RESEARCH-ARTICLE

TAXI: Traveling Salesman Problem Accelerator with X-Bar-Based Ising Macros Powered by SOT-MRAMs and Hierarchical Clustering

SANGMIN YOO, Oregon State University, Corvallis, OR, United States

AMOD HOLLA, Purdue University, West Lafayette, IN, United States

SOURAV SANYAL, Purdue University, West Lafayette, IN, United States

DONG-EUN KIM, Purdue University, West Lafayette, IN, United States

FRANCESCA IACOPI

DWAIPAYAN BISWAS, Interuniversity Microelectronics Centre, Leuven, Vlaams-Brabant, Belgium

View all

Published: 22 June 2023

DAC '25: 62nd Annual ACM/IEEE Design Automation Conference
June 22 - 26, 2023
California, San Francisco, United States

Conference Sponsors:
SIGDA

TAXI: Traveling Salesman Problem Accelerator with X-bar-based Ising Macros Powered by SOT-MRAMs and Hierarchical Clustering

Sangmin Yoo[1,2,4], Amod Holla[2], Sourav Sanyal[2], Dong Eun Kim[2],
Francesca Iacopi[1], Dwaipayan Biswas[3], James Myers[3], Kaushik Roy[2]
[1]*imec USA* & [2]*Purdue University*, West Lafayette, Indiana, USA, [3]*imec*, Leuven, Belgium,
[4]*Oregon State University*, Corvallis, Oregon, USA

2025 62nd ACM/IEEE Design Automation Conference (DAC) | 979-8-3315-0304-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/DAC63849.2025.11132522

***Abstract*—Ising solvers with hierarchical clustering have shown promise for large-scale Traveling Salesman Problems (TSPs), in terms of latency and energy. However, most of these methods still face unacceptable quality degradation as the problem size increases beyond a certain extent. Additionally, their hardware-agnostic adoptions limit their ability to fully exploit available hardware resources. In this work, we introduce TAXI – an in-memory computing-based TSP accelerator with crossbar(Xbar)-based Ising macros. Each macro independently solves a TSP sub-problem, obtained by hierarchical clustering, without the need for any off-macro data movement, leading to massive parallelism. Within the macro, Spin-Orbit-Torque (SOT) devices serve as compact energy-efficient random number generators enabling rapid "natural annealing". By leveraging hardware-algorithm co-design, TAXI offers improvements in solution quality, speed, and energy-efficiency on TSPs up to 85,900 cities (the largest TSPLIB instance). TAXI produces solutions that are only 22% and 20% longer than the Concorde solver's exact solution on 33,810 and 85,900 city TSPs, respectively. TAXI outperforms a current state-of-the-art clustering-based Ising solver, being 8× faster on average across 20 benchmark problems from TSPLib.**

***Index Terms*—Ising Machine, Accelerator, In-Memory Computing, Combinatorial Optimization Problem, SOT-MRAM.**

I. Introduction

Combinatorial optimization problems (CoPs), like the traveling salesman problem (TSP), are ubiquitous with applications in logistics, electronic design automation, genetics, and operations research [1], [2]. For TSPs, one of the most extensively studied CoPs, traditional methods [3] use heuristics like branch-and-bound and backtracking to find optimal solutions. However, such methods slow down exponentially as the problem size increases, due to exponential search space explosion. Additionally, traditional von Neumann architecture causes memory bottlenecks, rendering the TSP solvers inefficient for large instances due to repeated read-write operations during search-space exploration. This operating mismatch between the traditional TSP solver's execution pattern and von Neumann architecture's dataflow further slows down the optimization process, with the increase in problem size.

Ising models, inspired by statistical physics, offer a promising alternative for solving small-scale TSPs by encoding combinatorial state-spaces onto 2D arrays of binary spin values, that naturally settle into minimal energy states. Small

This work was funded through the joint MOU between the Indiana Economic Development Corporation, Purdue University, imec with in-kind support from the Applied Research Institute. Yoo participated in this work as a postdoctoral researcher at imec & Purdue University

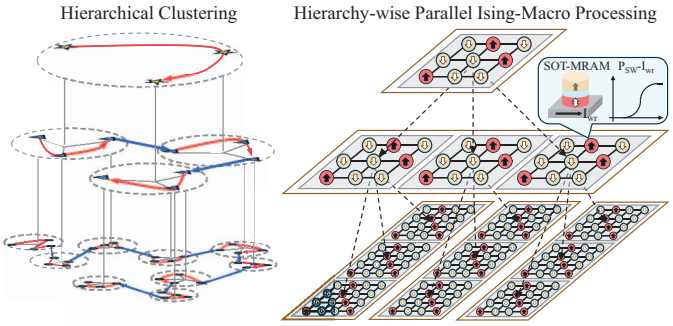

Fig. 1. Hierarchical clustering and Ising macros for large-scale traveling salesman problems. The shortest route within each cluster (red in the left) is optimized by an Ising macro (each grey box in the right). The final solution is derived by merging inter-cluster (blue in the left) and intra-cluster routes.

TSP instances can be efficiently mapped onto Ising crossbar (Xbar) arrays, potentially as hardware macros, enabling high-speed optimization. However, as the problem size grows, solution quality degrades due to the expanding search space and quadratic rise of spin interactions leading to a growing complexity of interconnects required to realize larger Ising Xbars. To address this issue, hierarchical clustering methods have been introduced [4]–[7], where large TSPs are decomposed into sub-problems that can be solved using smaller Ising models with acceptable quality. This work builds upon these clustering-based approaches, scaling further to tackle even larger problems with improved solution quality and further improvements in latency.

Typically, Ising solvers rely on stochastic processes to find near-optimal solutions. In conventional Complementary metal-oxide-semiconductor (CMOS)-based systems, random number generators (RNGs) are used to perform the stochastic switching of spins. However, these RNGs consume significant area and are too slow to meet the demands of rapid annealing, causing CMOS-based Ising solvers to be bulky with large area overheads and sluggish response-times [8], [9]. To overcome this, Spin-Orbit-Torque Magnetic Random Access Memory (SOT-MRAM)-based RNGs are considered in this work for Ising annealing, due to their energy-efficient and compact RNG implementation capability. In addition, in-memory computing (IMC) enables computations within memory arrays, reducing data movement and energy consumption. Xbar-based Ising primitives efficiently map TSP sub-problems onto hardware arrays [6], [7], making them suitable for realizing IMC macros. However, existing IMC-based Ising implementations still rely on off-macro memory accesses to store spin states.

To that effect, we propose TAXI – a TSP accelerator that uses Xbar-based Ising primitives as IMC macros with minimal off-macro data movement. TAXI applies hierarchical clustering to decompose large TSPs into smaller sub-problems, which are then solved in parallel using dedicated Ising hardware units. This architecture ensures that each sub-problem is solved directly in memory. This results in a hardware-algorithm co-design framework, where the algorithm (hierarchical clustering) and the architecture (spatially arranged Ising cores) are tightly integrated (See Fig 1). The custom hardware is designed by considering the algorithm's hierarchical execution pattern, ensuring that the Ising computations align with the algorithm's flow, enabling TAXI to maintain high solution quality as well as high speed even when problem size increases. Our contributions are as follows:

- A custom multiplication-and-accumulation (MAC)-based TSP solver using SOT-MRAMs and its stochastic switching to accelerate Ising annealing, overcoming the limitations of traditional CMOS-based RNGs (Section III).
- A novel hierarchical clustering with fixed inter-cluster routes that reduces TSP scale, with independently invoked Ising sub-solvers to enhance parallelization and sub-solution quality, enabling near-optimal solutions for large-scale TSPs (Section IV).
- A crossbar (Xbar)-based parallel architecture that supports in-macro Ising computing through hardware-algorithm co-design, mapping sub-problems onto Xbar arrays, and aligning the custom hardware with the algorithm to improve scalability (in terms of latency and energy) for large-scale TSPs (Section V).

Our experimental results (Section VI) demonstrate that TAXI is 8× faster on average across 20 benchmark problems (with up to 85,900 cities) from TSPLib [10] compared to a state-of-the-art hierarchical Ising solver [5]. TAXI generates 3% shorter traveling route on 33,810 cities over method [7] and 31% shorter route on 85,900 cities over method [5] – two recent clustering-based Ising works which demonstrate the best solutions on the respective problems, so far.

II. BACKGROUND AND RELATED WORKS

A. *Ising Solver*

In Ising model, Hamiltonian (energy) of the total system (H_{total}) and each spin (element) (H_i) are defined as below:

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

$$H_i = \sum_j J_{ij}\sigma_j + h_i \tag{2}$$

where J_{ij} represents interaction between spins, σ_i and σ_j, while h_i denotes external field applied to spins. Equation 1 can be reformulated by equation 2 as below:

$$H_{total} = -\sum_i \left(\sum_j J_{ij}\sigma_j + h_i \right) \sigma_i \tag{3}$$

According to the rule that spin state, +1 or -1, follows the polarity of H_i, greater than 0 or not, respectively, ΔH_{total} is *always negative*. It means every spin update minimizes H_{total}, approaching an optimal solution in an unsupervised manner.

Inspired by the nature, Ising solvers have been proposed to naturally find the optimal solution to problems that can be properly encoded in Ising format. For Ising acceleration, MAC operation based on a memory Xbar has been deployed [7], [11] for $\sum_j J_{ij}\sigma_j$ in equation 2; Spin states (σ_j) are applied to rows and multiplied with J_{ij}. Then, H_i is derived.

However, the always-descending energy update is insufficient to get the optimal solution when problems are complicated. The complication makes the energy search space of Ising problem non-monotonic as depicted in (Fig 2), causing the Ising solver to get stuck in local minimas. To climb up the energy hill and reach global minima which is optimal solution, Ising solver needs the power to violate the always-descending nature. For the violation, stochastic switching of spin disregarding equation 3 is necessary.

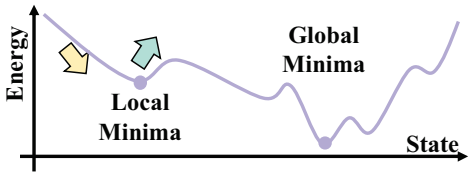

Fig. 2. Non-monotonic energy search space. Energy minimization (yellow) and Stochastic update (green) jointly find the global minima by enabling descending the energy landscape and escaping from local minimas, respectively.

B. *Source of Stochasticity*

CMOS-based random number generator (RNG) [7], [12], [13] and emerging electronic devices, such as resistive random access memory (RRAM) [11], silicon-oxide-nitride-silicon (SONS) FinFET [6], and spin-transfer torque MRAM [14]–[18] have been deployed as the source of stochasticity.

RRAM and SONS FinFET offer stochasticity through their inherent thermal and temporal noise when used for a Xbar array for MAC operation [6], [11]. Despite their better chip density, the intrinsic noise becomes uncontrollable when the Xbar grows [11]. Magnetic tunnel junction (MTJ)-based RNGs based on low barrier nanomagnets and telegraphic switching [15]–[18] and CMOS-based RNGs [8], [9] can be placed in the peripheral circuitry, which are not scaled up by Xbar size. However, current MTJ-based implementations require very low barrier nanomagnets ($\Delta \sim 0kT$) for truly high-speed RNG (>1Gb/s) and high-frequency sense circuitry to detect rapid switching of the device, while CMOS-based RNGs are slower (<2400Mb/s) [9] and take much more area ($> 375\mu m^2$) [8].

C. *Ising Solvers for Large-scale TSP*

Large-scale TSPs are hard for Ising solvers due to the quadratically growing number of connections as TSP size increases. Hierarchical Vertex Clustering (HVC) [4] proposed hierarchical clustering to reduce the number of connections. Despite the reduction, it still sparsely maps all the clustered problems to a single matrix, which leaves the connection problem unsolved when the problem size is beyond a certain

level. Neuro-Ising [5] proposed a hybrid clustering-and-graph-neural network manner but shows performance degradation as the problem size increases. In-memory annealer (IMA) [6] and its follow-up, digital compute-in-memory annealer (CIMA) [7], based on the hierarchical clustering achieves acceleration by simultaneously optimizing non-adjacent clusters. However, they store the spin states outside macros, causing slow-down.

III. SOT-MRAM-BASED TSP SOLVER

A. *MAC-based Energy Minimization for TSP*

TAXI leverages the Xbar-based MAC operation to minimize H_{total}. To achieve the objective of TSP, finding the shortest route to visit all cities only once, information on the distance between cities and the visiting order initialized by input order are used. The distance is reformulated as follows:

$$W_D(A,B) = \frac{D_{min}}{D_{A-B}} \times B_{Precision} \tag{4}$$

where $W_D(A,B)$ denotes a conductance value to map on a cross-point of Xbar, while D_{A-B} and D_{min} represent the distance between cities A and B and the minimum distance among all paths, respectively, and $B_{Precision}$ is the bit-precision that can be offered by the hardware.

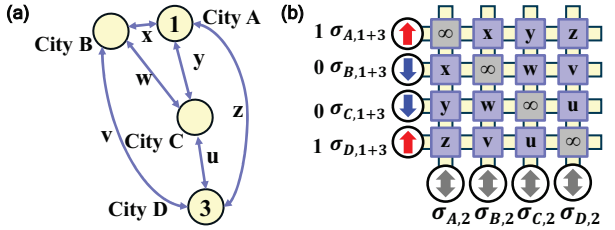

Fig. 3. (a) TSP with 4 cities. $u-x$ represent distances and numbers in a circle denote visiting orders. (b) Distance matrix mapped on a crossbar. Distances reformulated to W_D are programmed to crosspoints in resistance.

Fig 3 displays how the W_D is mapped on the Xbar and used to choose a city to visit at order *i* which results in the shortest route. For the optimization, cities visited at the previous (*i*-1) and the next (*i*+1) orders are considered because the total traveling distance is decided by both. The visiting information ($\sigma_{A,i}$) to a specific city (*A*) at a specific order (*i*) is represented by a binary spin, 1 or 0, following quadratic unconstrained binary optimization (QUBO) and Ising equivalence [20]. $\sigma_{A,1}$ and $\sigma_{D,3}$ are 1 because they are visited at order 1 and 3, respectively. With the fact that the distance information is static, we superpose the vector representing visiting information of order *i*-1 ($\sigma_{A-D,1}$) and *i*+1 ($\sigma_{A-D,3}$), as shown in Fig 3b. It reduces the crossbar array sizes, leading to better area-efficiency and less involvement of non-ideality.

In this setup, the current flowing through each column ($\sigma_{A-D,2}$) following Ohm's law and Kirchhoff's current law represents relative distance when the city is chosen for the order. It can be expressed as:

$$D_{x,i} = \sum_{k \neq x} W_D(x,k)(\sigma_{k,i-1} + \sigma_{k,i+1}) \tag{5}$$

According to equations 4 and 5, the current flowing through the 3rd column becomes the largest due to shorter distance (lower resistance), making $\sigma_{C,2}$ selected as up-spin.

B. *Stochasticity-Involved Decision Making*

In TAXI, stochasticity is involved in the choice of the largest current via a stochastically produced binary vector as big as the problem size. Only cities corresponding to 1s in the vector are considered for the choice. It offers the power to violate the always-descending nature, mentioned in Section II-A. The visiting order is updated by the newly chosen city.

C. *Ising Macro as a TSP Solver*

A Xbar array with customized peripheral circuitry is leveraged as an Ising macro to solve a TSP in the manner of the aforementioned algorithms. Fig 4 shows the system-level layout of the Xbar-based Ising macro along with the associated peripherals inset in Fig 4c,d. The unit cell of the Xbar array is a 3T-1M SOT-MRAM unit, with a transmission gate to control the read current, and 1 transistor to control the write current. The transmission gate was chosen owing to the need for accurate bi-directional reads.

The Xbar is divided into $B+1$ partitions, where B is the chosen bit-precision for the W_D. The first B partitions contain the W_D, while the last partition stores the spin information ($\sigma_{k,i}$) as the solution of the given TSP. To minimize effect from non-ideality, such as wire resistance, higher significant bit is stored closer to the left end. The rows and columns of the spin storage (SS) represent cities and visiting orders. If a specific city (A) is chosen at a specific order (1), a SOT-MRAM that is located in the first row and the first column in the SS is programmed in the low resistance state (LRS), while the others on the same column are in the high resistance state (HRS). This implies that the size of the Xbar required for a TSP problem size of N is $N \times N \cdot (B+1)$.

1) ***Superpose****:* The superposition of vectors representing visiting information, discussed in Section III-A and Fig 3, is executed by MAC operation using the SS, as illustrated in Fig 4a. In the case of Fig 3, the first and the third columns are activated to recall visiting information at order 1 and 3. Then, the Xbar natively derives the superposed vector in current at the end of the rows, while a following current comparator [21] translates it in binary and a D-latch stores the binary vector. This flow is depicted by the red arrow in Fig 4.

2) ***Calculate Distance****:* The stored vector in the D-Latch is fed back to the rows, generating currents by executing equation 5, based on the W_D stored in the first B partitions of the Xbar. Only columns in the first B partitions are activated, with columns in the SS deactivated. The currents from the Xbar flow into current mirrors, scaled by 2^{b-1} depending on the bit-significance (b) of the partition, as shown in Fig. 4b.

3) ***Stochastic Binary Vector*** *:* The stochastic binary vector mentioned in Section III-B is generated by SOT-MRAM's stochastic switching-based circuit consisting of N identical units shown in Fig 4c. SOT-MRAM has two ferromagnetic layers (one with a fixed magnetic moment direction, while another is free to rotate) separated by a thin insulator. Its resistance is determined by how many electrons can tunnel through this MTJ, which depends on whether the magnetic moment of the

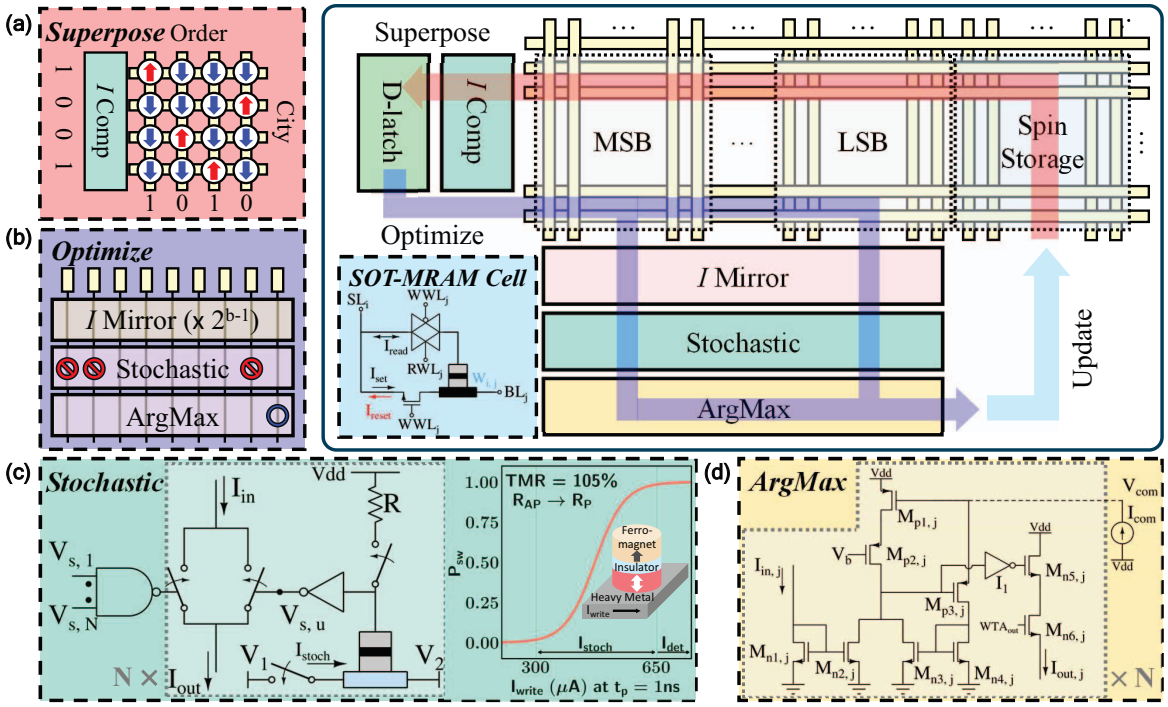

Fig. 4. Floor plan of an Ising macro. Arrows represent data flow across the macro. Insets (a-d) illustrate components in the macro. (a) Superposition of vectors to optimization. Red and blue arrows represent up- and down-spin stored in spin storage, respectively. (b) Optimization by crossbar array as the distance matrix and following peripheral circuits. (c) SOT-MRAMs in the stochastic circuit pass or prevent current from the crossbar following its stochastic switching. Inset displays the probability of switching R_{AP} to R_P of the chosen SOT device [19]. I_{stoch} and I_{det} denote the current ranges for stochastic and deterministic operations, respectively. (d) ArgMax circuit picks a city to visit by choosing the largest current.

two ferromagnetic layers is aligned in parallel (R_P) or anti-parallel (R_{AP}). This alignment can change due to the Spin Hall effect induced by current flowing through a heavy metal layer under the ferromagnet [22]. The stochastic switching that TAXI leverages is based on this dynamic. Depending on the amount of current, the probability of switching (P_{sw}) between R_P and R_{AP} can be controlled, as shown in Fig 4c [19]. By controlling the current that derives a specific P_{sw} (300μA — 650μA), TAXI can control the expected number of 1s in the stochastic binary vector. SOT-MRAMs in the Xbar are operated only in the deterministic regime (>650μA).

The stochastic circuit (Fig. 4c) controls the flow of the preceding current mirror outputs into the next stage upon the stochastic switching. For a given unit, if the SOT device switches, the voltage across the divider circuit becomes lower than units whose SOT device has not switched. The output of the inverter is therefore high, resulting in the unit allowing the input current to pass through. The circuit allows the current only through the columns ($\mathcal{S}$) where the SOT device has stochastically switched for a given iteration to pass through. In case $\mathcal{S} = \varnothing$, currents through all columns are allowed to pass because the NAND gate output is high for every unit.

*4) **Choosing the Largest Current**:* The largest current is selected by the ArgMax circuit shown in Fig 4d. It processes a vector of input currents and outputs a vector where only the index 'i' corresponding to the largest input current remains nonzero. The winner current is set to the minimum required to switch a SOT-MRAM device in the deterministic regime. The ArgMax circuit is based on Lazzaro's winner-take-all circuit [23], enhanced with a cascaded transistor ($M_{p2,j}$) in the input branch to increase output resistance [24], and a current mirror ($M_{n2,j}$, $M_{n3,j}$) to boost feedback, significantly improving resolution and speed [25].

*5) **Update Spin Storage**:* The one-hot encoded current vector derived by the ArgMax circuit is directly used to update the spin storage, as illustrated by the cyan arrow in Fig 4. Before the update operation, all the devices on the column representing the order which is being optimized are reset to HRS which represents 0. Then, the output current vector of the ArgMax circuit is applied to write the column.

*6) **Annealing**:* The processes described in sections III-C1 - III-C5 are repeated for the pre-designated number of iterations from the first to the last order by activating different columns in the SS depending on the visiting order to optimize. After the whole iteration, the spin information stored in the SS is retrieved as the final solution of the given TSP. To optimize the solution, stochasticity should decrease during iterations, allowing the Ising system to remain at the global minimum as the solution improves. Otherwise, the system may escape the global minimum and end up with a worse solution. The solution is optimized the best when the stochasticity decreases slowly. Optimization is however faster in the early stages making the non-linear decrease of stochasticity appealing.

For the non-linearity, TAXI utilizes the native sigmoidal switching probability vs I_{write} characteristic, as shown in Fig 4c. The non-linearity can enable a relatively rapid decrease in stochasticity at the earlier stage, leading to shorter overall latency, while keeping the advantage of slow decrease at

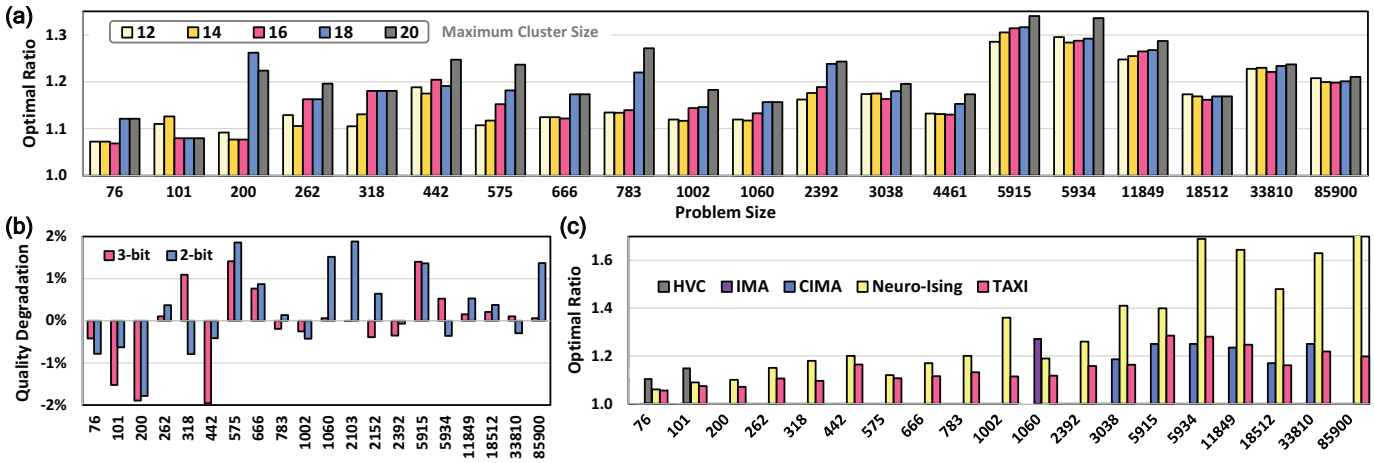

Fig. 5. (a) Optimal ratio depending on the maximum cluster size in 4-bit precision. (b) Solution quality degradation when the bit precision changes from 4-bit to lower-bit options. The positive number represents degradation. (c) Comparison of solution optimality. Data of TAXI with 4-bit precision and 12 cluster size are presented. Data of other Ising solvers are adapted from [5]–[7].

the later stage. To leverage the native sigmoidal decrease, the I_{write} is initialized to 420μA (corresponding switching probability, 20%) and linearly decreased by 50nA after every iteration. Once the I_{write} becomes 353μA (corresponding probability, 1%), the Ising solver stops, then, the solution stored in the SS is taken as the final output of the given TSP.

IV. HIERARCHICAL CLUSTERING

*1) **Locally Solve Global Problems**:* We adopt a hierarchical structure similar to HVC [4] for clustering, executed in a bottom-up fashion. After clustering all cities at level i, the cluster centroids form level $i+1$. This process repeats until the number of centroids at a level is less than the maximum TSP size confidently solvable by an Ising macro. The number of clusters at each level is determined by the maximum size.

*2) **Parallel Processing**:* After the hierarchical structure is defined, the TSP is solved in a top-down manner. Once the topmost TSP (level L) is solved, the solution, which is the visiting order of clusters at level $L-1$, is determined. Unlike HVC, which sparsely maps all clusters to a single Xbar array to optimize routes both within and between clusters, TAXI fixes the first and last cities of sub-problems first to minimize the inter-cluster route. This fixation ensures that sub-problem solutions do not compromise the inter-cluster distance. The first and last cities of each cluster are determined by identifying the closest city pairs between neighboring clusters, enabling parallel solving of as many clusters in a level as there are Ising macros in a chip. The entire process concludes when all sub-problems in the lowest level are solved.

*3) **Agglomerative Clustering**:* TAXI leverages Agglomerative clustering to achieve the hierarchical clustering for its robustness to outliers, instead of K-means clusters adopted by others [4]–[7]. It is a type of hierarchical clustering that merges clusters based on an inter-cluster similarity metric until the desired number of clusters is obtained [26]. With the Ward linkage, the squared distance or variance within a cluster is minimized [27]. Even though K-means clustering also aims to minimize the intra-cluster variance, they usually lead to nearly spherical or regular clusters [28], while agglomerative clustering can produce compact irregular clusters.

V. X-BAR-BASED PARALLEL ARCHITECTURE

Hierarchical clustered Ising macros are mapped to the Xbar architecture to enable in-macro Ising computing through hardware-algorithm co-design, mapping sub-problems onto Xbar arrays, and aligning the custom hardware with the algorithm to improve scalability for large-scale TSPs. We instrumented the PUMA architecture [29] to perform the mapping and evaluated the latency and energy consumption from data movement between off-chip memory and the proposed Ising macros executing parallel Ising computations. PUMA is a cycle-accurate simulator based on an IMC spatial architecture, structured with a hierarchy of chip, tile, core, and MVMUs. The simulator's compiler generates instructions using PUMA ISA, and the simulator executes them to assess latency and energy. We replaced PUMA's ReRAM-based MVMUs with our Ising macro and scaled all elements from PUMA's 32nm to a 65nm node. See Section VI-C for more details.

VI. EVALUATION

A. *Results on Multiple TSPs*

The performance of TAXI with varied maximum cluster sizes is simulated in C++, following the Ising layer presented in [5]. For realistic simulation, ON/OFF resistance of SOT-MRAM and transistors and wire resistance are considered [19]. Fig 5a displays the optimal ratio which is derived by dividing a solution of Ising solver with exact solution [30]. A smaller cluster (sub-problem) size leads to a better solution quality in most cases as Ising solvers usually show better results on smaller problems. With the problem size of 12, we tested different bit precisions for the W_D.

Fig 5b presents solution quality degradation when the bit precision changes from 4-bit to 3- or 2-bit. It shows that the performance is maintained in the 2% range. We attribute this fluctuation to variation of both W_D precision and non-ideality arising from Xbar size change. From the energy-efficiency

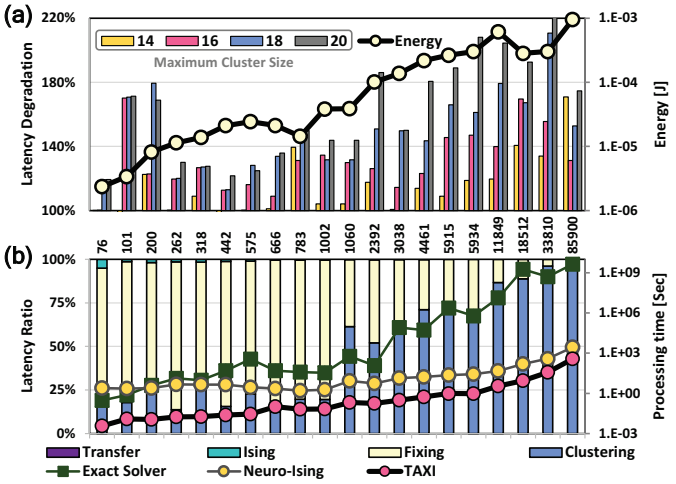

Fig. 6. (a) Latency (bar) and energy (line) arising from Ising macros and data transfer within the architecture depending on the maximum cluster size in 4-bit precision. Energy based on a problem size of 12 and 2-bit precision is representatively displayed. (b) Total Latency including clustering, fixing, Ising processing, and data transfer illustrated in lines (right y-axis). Bars represent the ratio of each component contributing to the total latency (left y-axis).

TABLE I
CIRCUIT SIMULATION RESULTS FOR COMPLETION OF 1 ITERATION INCLUDING SUPERPOSITION, OPTIMIZATION, AND SPIN-STORAGE UPDATE.

		2 bit	3 bit	4 bit
Array Size		12×36	12×48	12×60
Power [mW]		4.202	5.033	5.11
Latency [ns]	**Superposition**	3	3	3
	Optimization	4	4	4
	Storage Update	2	2	2
Energy [pJ]		37.82	45.3	45.98

perspective, the lower bit-precision is beneficial. Depending on the priority between the power budget and the solution quality, TAXI can be reconfigured. The comparison will be discussed in the following section VI-C. TAXI with a clustering size of 12 and 4-bit precision is benchmarked with existing Ising solvers for TSPs in Fig 5c. TAXI outperforms the others [5]–[7] in most cases including three of the largest TSPs, 18512, 33810, and 85900.

B. Circuit Simulation

An entire Ising macro shown in Fig 4 is simulated using the `Cadence® Spectre®` simulator in the TSMC 65nm technology node. Verilog-A behavioral modeling was performed to fit the characteristics of the considered SOT device in section III-C3. We choose a TSP problem size of 12. We run the simulation for 1 complete iteration including the superposition, optimization, and spin-storage updates. The pre-layout circuit simulation results are presented in Table I. The higher-bit precision causes a larger area and higher energy consumption.

C. Architecture Simulation

The total latency excluding clustering with fixing simulated by PUMA simulator is presented in Fig 6a. It presents the ratio of the latency with a specific maximum cluster size to that with a maximum cluster size of 12. The higher value means the slower operation. In most cases, the larger cluster size causes longer latency. The clustering time does not vary noticeably depending on the problem size. The latency arising from fixing the intercluster routes theoretically increases as the problem size increases, because the possible number of pairs between clusters increases quadratically while the number of clusters linearly decreases. Based on the results presented in Fig 5a and Fig 6a, the smaller problem size is preferred if the chip can contain as many Ising macros as the number of clusters in a level. It enables mapping and solving all clusters in parallel. Otherwise, the larger size could be a better option to maximize the parallel operation. The total energy consumption excluding clustering and fixing is presented in line in Fig 6a. Lower energy consumption for transferring, mapping, and processing (presented in Table I) smaller W_D makes the lower bit-precision more efficient. Table II shows the energy consumption, excluding data transfer and mapping, to fairly compare TAXI and previous works, demonstrating that ours is the most power-efficient.

TABLE II
ENERGY COMPARISON WITH STATE-OF-THE-ART

	[4]	[6]	[7]	**This Work**
Technology	CPU	14nm FinFET	16/14nm CMOS	65nm CMOS + SOT-MRAM
Problem Size	101	1060	33K, 86K	1060, 33K, 86K
Energy (J)	1.1	20.08μ	$\sim 20\mu, \sim 45\mu$	1.81μ, 2.67μ, $3.07\mu^{\dagger}$

† Excludes mapping energy. Including mapping, energies are $38.7\mu J$, $302\mu J$, and $952\mu J$ for TSPs with sizes 1060, 33K and 86K respectively.

Fig 6b shows the overall latency of TAXI with a maximum cluster size of 12 along with that of Neuro-Ising [5] and an exact solver [3]. We exclude CIMA [7] from this comparison as they do not report the total latency including clustering. The bar plots display the ratio of each component contributing to the total latency. As problem size increases, the speed advantage over the exact solver gets bigger and the total latency is dominated by clustering and fixing. The biggest problem, pla85900, is solved by TAXI in 375.4s and consumes 9.51×10^{-4}J, while it takes a projected time of 136 years (1.14×10^7 longer) and 3.82×10^{11}J (4.01×10^{14} more) for the exact solver on a single core CPU [31].

VII. CONCLUSION

We discussed existing hierarchical clustering-based Ising solvers suffer performance degradation with increasing problem size, due to search-space explosion and inherent mismatch between hierarchical execution pattern and hardware architecture data flow. To that effect, we proposed TAXI – a hardware-algorithm co-designed TSP accelerator with Xbar-based Ising macros – which outperforms existing Ising solvers on very large-scale TSPs. We report $8\times$ latency improvement across 20 benchmarks over a clustering-based Ising solver [5] from TSPLib [10], up to the largest TSP with 85,900 cities. TAXI is endowed with SOT-MRAM-based in-memory Ising primitives and a hierarchical clustering algorithm. The co-design enables TAXI to solve sub-problems of large-scale TSPs on a cluster-by-cluster basis without off-macro memory access, resulting in high inter-cluster parallelism with graceful acceleration, while maintaining solution quality within an acceptable $1.2\times$ that of exact solver [3] on a TSP with 85,900 cities.

REFERENCES

[1] MH BJORNDAL, A CAPRARA, PI COWLING, F DELLACROCE, H LOURENCO, F MALUCELLI, AJ ORMAN, D PISINGER, C REGO, and JJ SALAZAR. Some thoughts on combinatorial optimization. *EUROPEAN JOURNAL OF OPERATIONAL RESEARCH*, 83(2):253–270, JUN 8 1995.

[2] Jann Michael Weinand, Kenneth Sörensen, Pablo San Segundo, Max Kleinebrahm, and Russell McKenna. Research trends in combinatorial optimization. *International Transactions in Operational Research*, 29(2):667–705, 2022.

[3] David Applegate and William Cook. Concorde tsp solver, 2006.

[4] Akira Dan, Riu Shimizu, Takeshi Nishikawa, Song Bian, and Takashi Sato. Clustering approach for solving traveling salesman problems via ising model based solver. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[5] Sourav Sanyal and Kaushik Roy. Neuro-ising: Accelerating large-scale traveling salesman problems via graph neural network guided localized ising solvers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(12):5408–5420, 2022.

[6] Anni Lu, Jae Hur, Yuan-Chun Luo, Hai Li, Dmitri E. Nikonov, Ian A. Young, Yang-Kyu Choi, and Shimeng Yu. Scalable in-memory clustered annealer with temporal noise of charge trap transistor for large scale travelling salesman problems. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 13(1):422–435, 2023.

[7] Anni Lu, Junmo Lee, Yuan-Chun Luo, Hai Li, Ian Young, and Shimeng Yu. Digital cim with noisy sram bit: A compact clustered annealer for large-scale combinatorial optimization. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, New York, NY, USA, 2024. Association for Computing Machinery.

[8] Kaiyuan Yang, David Fick, Michael B. Henry, Yoonmyung Lee, David Blaauw, and Dennis Sylvester. 16.3 a 23mb/s 23pj/b fully synthesized true-random-number generator in 28nm and 65nm cmos. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 280–281, 2014.

[9] Sanu K. Mathew, Suresh Srinivasan, Mark A. Anders, Himanshu Kaul, Steven K. Hsu, Farhana Sheikh, Amit Agarwal, Sudhir Satpathy, and Ram K. Krishnamurthy. 2.4 gbps, 7 mw all-digital pvt-variation tolerant true random number generator for 45 nm cmos high-performance microprocessors. *IEEE Journal of Solid-State Circuits*, 47(11):2807–2821, 2012.

[10] Gerhard Reinelt. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[11] Fuxi Cai, Suhas Kumar, Thomas Van Vaerenbergh, Xia Sheng, Rui Liu, Can Li, Zhan Liu, Martin Foltin, Shimeng Yu, Qiangfei Xia, J Joshua Yang, Raymond Beausoleil, Wei D Lu, and John Paul Strachan. Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nature Electronics*, 3:409–418, 2020.

[12] Masanao Yamaoka, Chihiro Yoshimura, Masato Hayashi, Takuya Okuyama, Hidetaka Aoki, and Hiroyuki Mizuno. A 20k-spin ising chip to solve combinatorial optimization problems with cmos annealing. *IEEE Journal of Solid-State Circuits*, 51(1):303–309, 2016.

[13] Chieh-Pu Lo, Sirish Oruganti, Yipeng Wang, Mengtian Yang, Shanshan Xie, and Jaydeep P. Kulkarni. Data movement-aware, ping-pong ising machine supporting full connectivity and variable bitwidths. *2024 IEEE European Solid-State Electronics Research Conference (ESSERC)*, pages 721–724, 2024.

[14] Saima Sharmin, Yong Shim, and Kaushik Roy. Magnetoelectric oxide based stochastic spin device towards solving combinatorial optimization problems. *Scientific Reports*, 7, 12 2017.

[15] Akio Fukushima, Takayuki Seki, Kay Yakushiji, Hitoshi Kubota, Hiroshi Imamura, Shinji Yuasa, and Koji Ando. Spin dice: A scalable truly random number generator based on spintronics. *Applied Physics Express*, 7(8):083001, jul 2014.

[16] Won Ho Choi, Yang Lv, Jongyeon Kim, Abhishek Deshpande, Gyuseong Kang, Jian-Ping Wang, and Chris H. Kim. A magnetic tunnel junction based true random number generator with conditional perturb and real-time output probability tracking. In *2014 IEEE International Electron Devices Meeting*, pages 12.5.1–12.5.4, 2014.

[17] Hochul Lee, Farbod Ebrahimi, Pedram Khalili Amiri, and Kang L. Wang. Design of high-throughput and low-power true random number generator utilizing perpendicularly magnetized voltage-controlled magnetic tunnel junction. *AIP Advances*, 7(5):055934, 03 2017.

[18] Kerem Yunus Camsari, Sayeef Salahuddin, and Supriyo Datta. Implementing p-bits with embedded mtj. *IEEE Electron Device Letters*, 38(12):1767–1770, 2017.

[19] K. Cai, G. Talmelli, K. Fan, S. Van Beek, V. Kateel, M. Gupta, M.G. Monteiro, M. Ben Chroud, G. Jayakumar, A. Trovato, S. Rao, G.S. Kar, and S. Couet. First demonstration of field-free perpendicular sot-mram for ultrafast and high-density embedded memories. In *2022 International Electron Devices Meeting (IEDM)*, pages 36.2.1–36.2.4, 2022.

[20] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem, 2008.

[21] H. Träff. Novel approach to high speed cmos current comparators. *Electronics Letters*, 28:310–311, 1992.

[22] Luqiao Liu, O. J. Lee, T. J. Gudmundsen, D. C. Ralph, and R. A. Buhrman. Current-induced switching of perpendicularly magnetized magnetic layers using spin torque from the spin hall effect. *Phys. Rev. Lett.*, 109:096602, Aug 2012.

[23] J. Lazzaro, S. Ryckebusch, M.A. Mahowald, and C. A. Mead. Winner-take-all networks of o(n) complexity. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[24] B. Sekerkiran and U. Cilingiroglu. Improving the resolution of lazzaro winner-take-all circuit. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 2, pages 1005–1008 vol.2, 1997.

[25] J.A. Starzyk and X. Fang. Cmos current mode winner-take-all circuit with both excitatory and inhibitory feedback. *Electronics Letters*, 29:908–910, 1993.

[26] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[27] Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[28] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[29] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R. Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, and Dejan S. Milojicic. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 715–731, New York, NY, USA, 2019. Association for Computing Machinery.

[30] Concorde benchmarks (from 1999). https://www.math.uwaterloo.ca/tsp/concorde/benchmarks/bench99.html.

[31] Concorde benchmarks on pla85900. https://www.math.uwaterloo.ca/tsp/pla85900/compute/cpu.htm.