Fully Parallel Multi-Agent Photonic Optimizer

Ghazi Sarwat Syed,[1] Philipp Schmidt,[2] Frank Brückerhoff-Plückelmann,[2] Jelle Dijkstra,[2] Wolfram H.P Pernice,[2] and Abu Sebastian[1,2]

[1)] *IBM Research – Europe, Säumerstrasse 4, 8803 Rüschlikon, Switzerland*

[2)] *Kirchhoff-Institut für Physik, Im Neuenheimer Feld 227, 69120 Heidelberg, Germany*

SUPPLEMENTARY INFORMATION S1. PHASE CHANGE MEMORY BASED WEIGHTS

In the manuscript, we discussed that an ideal IPO unit would take advantage of matrix stationarity by keeping computation and data in the same place, avoiding traditional separate processing. In such an approach, the matrix elements can be stored in the non-volatile states of photonic phase-change memory (PCM) cells, which sit on top of the connecting waveguides. In the matrix implementation, the PCM cells therefore act as attenuating matrix elements that absorb a desired amount of light depending on their phase configuration (Nature 589, 52–58 (2021)). In the following, we use optical pulse parameters and compute precision demonstrated by PCM-based photonic integrated circuits in Ref (Science advances 5, eaau5759 (2019)), making use of narrow-band light sources, to simulate the IPO. For encoding negative numbers in the input vectors, differential encoding (i.e., positive and negative columns of the crossbar) is assumed. We also simulate the IPO using standard algorithms for evolutionary and swarm optimization.

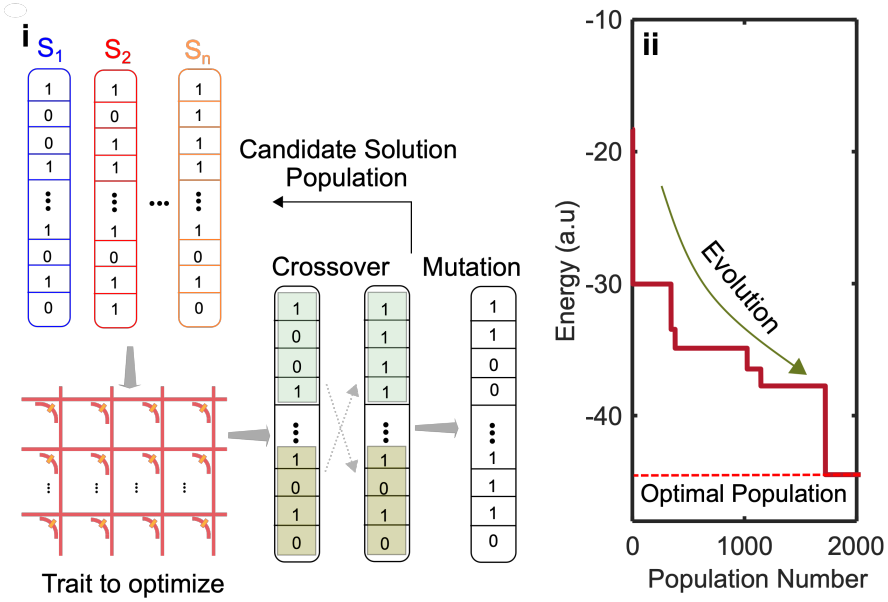


FIG. S1: **Combinatorial Optimization Problems**. Here, we present the approach of applying inversions to the nodal distributions for the MaxCut problem using the standard Genetic Algorithm (Parallel computing 7, 65–85 (1988), Multimedia Tools and Applications 80, 8091–8126 (2021)). (i) This evolutionary algorithm implements the process of natural selection, i.e., survival of the fittest among individuals of successive generations to solve a given problem. We apply this to the Max-Cut problem on the cyclic graph discussed previously. In our implementation, the algorithm maintains a population P_k of n individuals, encoded into optical query vectors s_n, where k denotes the index of the evolving population. The elements of a query vector represent the chromosomes (sets of genes), and the quality or fitness of an individual is measured using a fitness score. (ii) We map the fitness score to the Hopfield energy. The goal is that, within a given population P_k, individuals with higher fitness scores are given more opportunities to reproduce. Reproduction includes mutations and crossovers, applied using swap ($\bigoplus_s$) and flip ($\bigoplus_f$) operators. In our implementation $\bigoplus_s$ is applied between the two best-performing queries, swapping genes to generate new queries. $\bigoplus_f$ is applied by randomly flipping genes. These operators provide best-case inversions, leading to better-performing individuals and, therefore, a stronger population overall.

SUPPLEMENTARY INFORMATION S2. FURTHER DISCUSSION ON LINEAR PROGRAMMING IMPLEMENTATION

In Figure 5 of the main text, we presented our implementation of the canonical particle swarm optimization. In Figure S3, we provide additional data illustrating the convergence behavior as a function of both the number of agents and the noise amplitude.

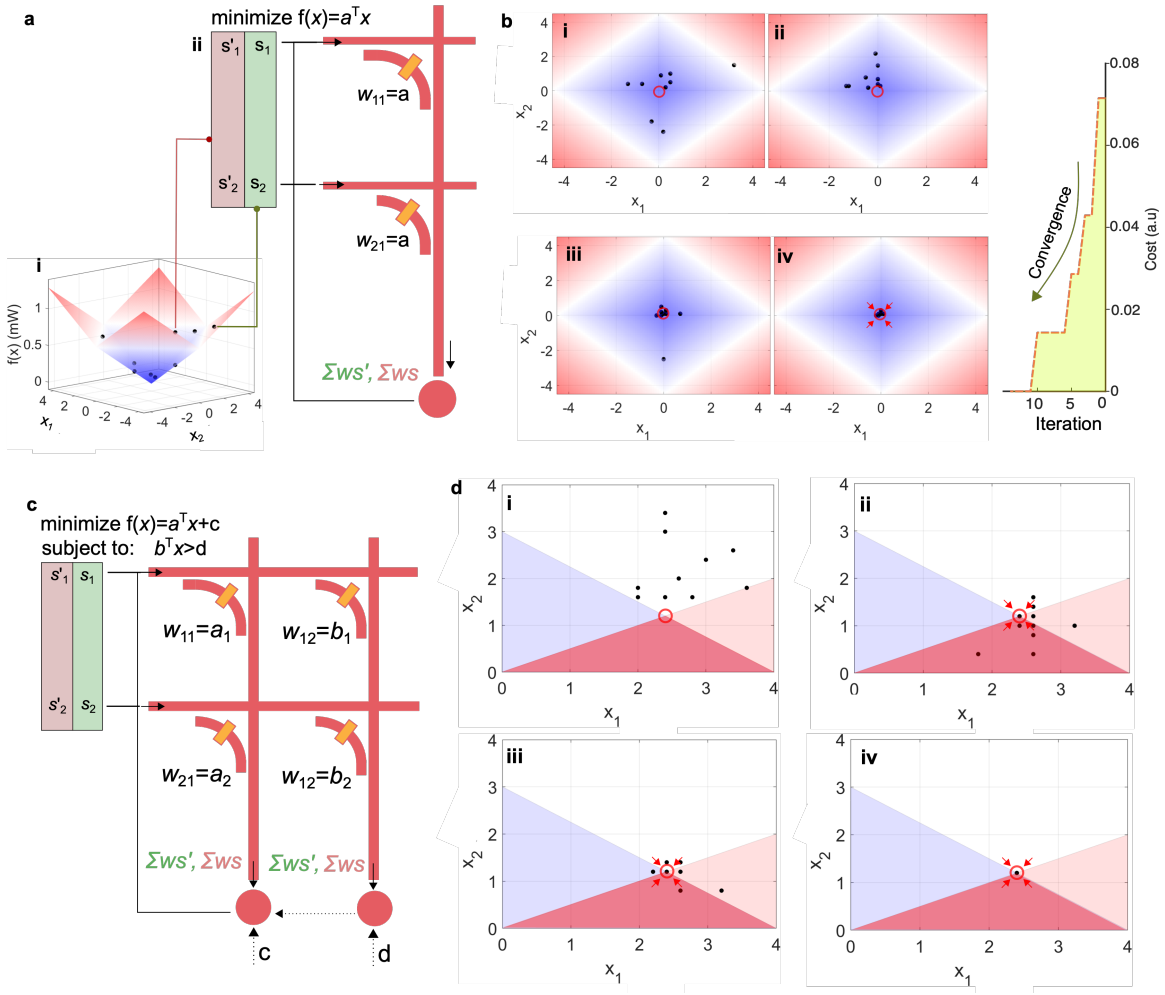


FIG. S2: **Linear Programming Problems**. (a) A schematic showing the configuration of a crossbar unit used for solving a continuous optimization problem ($|x_1 + x_2|$) through a swarm optimization scheme. Here, the standard PSO algorithm is used. The algorithm is reviewed in detail in the references (IEEE Cat. 01TH8546, 1, 81–86 (2001), Soft computing 22, 387–408 (2018)). Briefly, the agent s_c^t provides a means to generate the future query s_c^{t+1} through combination of the objective function computation and feedback from other agents. This follows $s_c^{t+1} = s_c^t + \vec{d^{t+1}}$. $\vec{d^{t+1}} = \vec{d^t} + f([\alpha, p_{best}], [\gamma, g_{best}])$, where p_{best} and $F = g_{best}$ represent the best solutions for an agent through its personal (and previous) search, and the best solution by an agent in the population, respectively. α and γ are scalars which provide a mechanism to control the updates. (b) Multiple agents encoded into the amplitude modulated optical signals operate in parallel to perform exploration and exploitation on such a unit. (i-iv) Snap shots taken at different time stamps showing iterative convergence of the queries to the optimal solution. In each iteration, the best query guides other queries to convergence (red dots), while maintaining the queries' individualistic exploration behavior. The right panel shows the minimization of cost function during convergence. (c) A sketch showing the configuration of a crossbar unit used in solving constrained optimization problems. The objective and constraint functions are mapped onto the photonic crossbar and solved simultaneously by using parallel optical signals. (d) Snap-shots taken at different time stamps during solving such a problem, showing the convergence of the agents to the optimal solution.

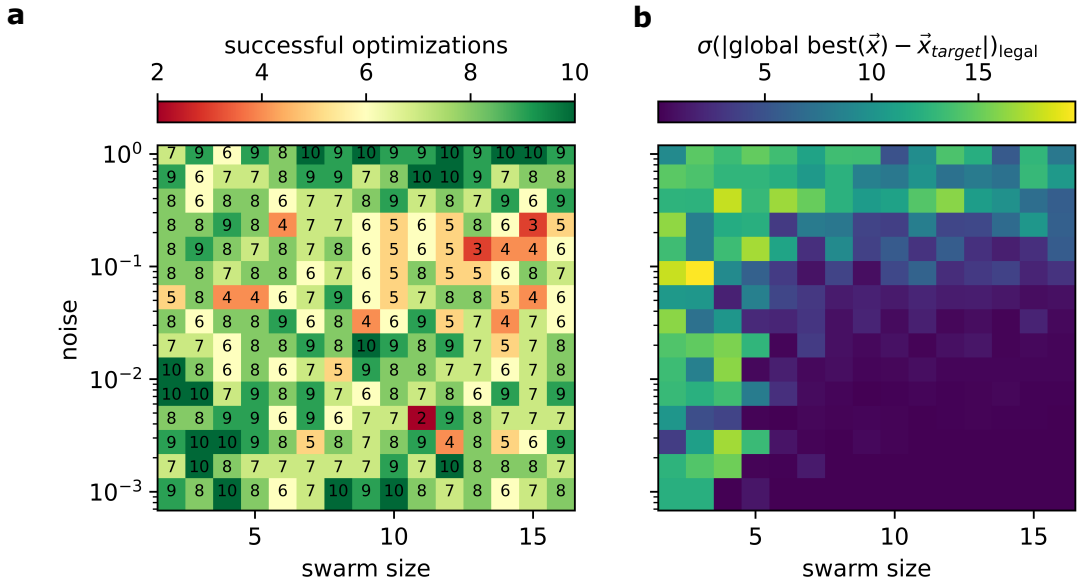


FIG. S3: **Success rate and stability of the PSO solutions (a)** Number of successful runs (out of 10 PSO optimizations) for each noise and swarm size combination that produced at least one legal minimum. **(b)** Standard deviation of the distance to the actual minimum across all legal runs from the noise vs. swarm size sweep shown in Figure 5c, highlighting unstable solutions at high noise levels and low particle counts.

SUPPLEMENTARY INFORMATION S3. COMPUTE PRECISION

The numeric precision of compute depends particularly on whether the problem is discrete or continuous. In discrete optimization, variables take values from a finite set (often binary). After performing a MVM, the outputs can be binarized by thresholding operation o produce discrete variable values. Small errors in the MVM are tolerated as long as the sign (threshold crossing) is correct, making the operation robust to limited bit-width. In this context, thresholding dominates, and compute precision only needs to be sufficient to determine the correct threshold (US Patent 12,422,882.). For all outputs $i = 1,\ldots,m$; $\hat{y}_i = f_{\text{th}}(\tilde{y}_i) = f_{\text{th}}(y_i + \varepsilon_i)$. This output is correct as long as $|\varepsilon_i| < |y_i - \theta_i|$, where $f_{\text{th}}(\cdot)$ is the thresholding function.

In contrast, continuous optimization involves variables that take values from a continuous domain. Here, small errors in MVM directly propagate into the solution, and high compute precision is critical to achieve accurate convergence. Continuous problems are therefore more challenging to solve with low-precision or noisy hardware. Our goal was to demonstrate that, despite these challenges, it is possible to apply our scheme to more difficult continuous optimization problems and still achieve reasonable convergence. This approach enables a warm-start strategy, providing an initial good solution that can subsequently be refined using other higher-precision or auxiliary solvers for further convergence. In another approach it also possible to improve the compute precision of our hardware by using coherent light sources, such as from frequency combs, and through noise-injection at the outputs. We are currently exploring this approach.

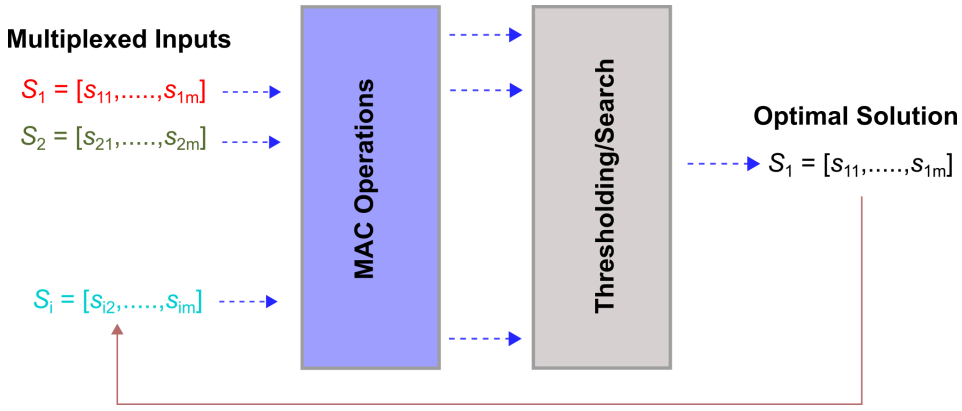


FIG. S4: **Iterative Solver**. The proposed photonic optimizer operates as an iterative solver. In solving optimization problems, particularly combinatorial problems, the system repeatedly updates the state through a feedback loop. The feedback signals, as well as dynamic state updates, can be implemented using high-bandwidth thresholding logic operations (e.g., XOR and MUX) applied against a reference vector that encodes the required thresholds.

In Fig. S5, we present additional results on the achievable MVM compute precision in our setup. In our experiments, amplified spontaneous emission ASE is the primary source of noise, arising from both phase and intensity fluctuations. This noise follows a white, uncorrelated profile, resulting in a Gaussian distribution. To demonstrate that the noise profile can be controlled, we apply averaging over multiple outputs, which leads to progressively tighter distributions. We note that alternative noise injection mechanisms can also be employed, for example within the crossbar or at the output stage, thereby obviating the need for averaging cycles that incur latency overheads. In such schemes, employing a deterministic source also facilitates the solution of continuous optimization problems, where the level of precision needed is higher.

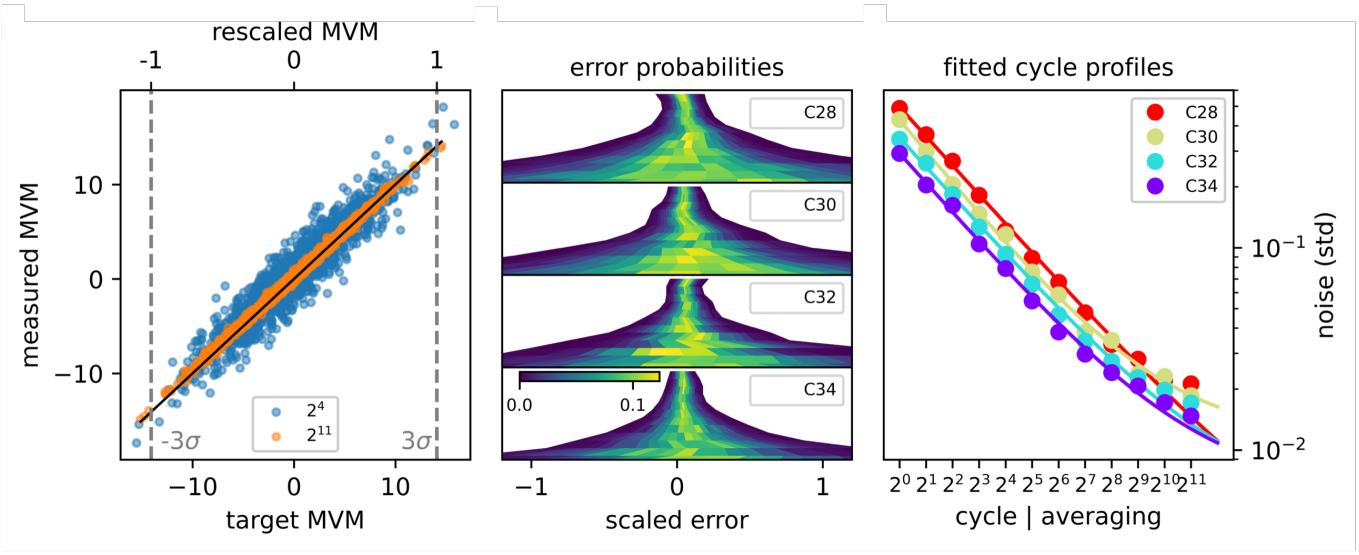


FIG. S5: Numerical Precision of MVMs. The left panel shows exemplary random MVM distribution under averaging of 2^4 and 2^{11}. The raw data is normalized such that the interval [-1, 1] corresponds to $[-3\sigma, 3\sigma]$. The middle panel illustrates the resulting scaled error distributions across the 12 measured averaging factors for each channel. Despite minor deviations the simulations assume a gaussian distribution with mean 0. The right panel shows the resulting normalized measured noise profiles for all four channels under different averaging values with their fits. For the MAX-cut solver, the number of averages directly corresponds to the cycle index.

SUPPLEMENTARY INFORMATION S4. ARCHITECTURE ANALYSIS

Performance Metric:

Using the parameters of the present demonstrator system, we estimate an overall efficiency of 0.022 TOPS/W. At this stage, the dominant contribution to power consumption arises from the supporting electronic components, in particular the digital-to-analog converters (300 mW per DAC), analog-to-digital converters (270 mW per ADC), and transimpedance amplifiers (3 W per TIA). Therefore, this is an upper limit descrbining a more sophisticated system since for discrete optimization problems the 1-bit input and output encoding simplifies these overheads. Furthemore, the the overall efficiency depends strongly on the size of the photonic accelerator. For an $N \times N$ matrix operation, the number of converters and TIAs scales linearly with N, whereas the number of computational operations scales as N^2. As a result, the effective efficiency is expected to improve with increasing system size. This favorable scaling is one of the main reasons why photonic accelerators remain promising despite the current electronic overhead. Recent analyses have also highlighted the potential for substantially improved energy efficiency in larger-scale photonic systems (arXiv:2511.00186 (2025)).

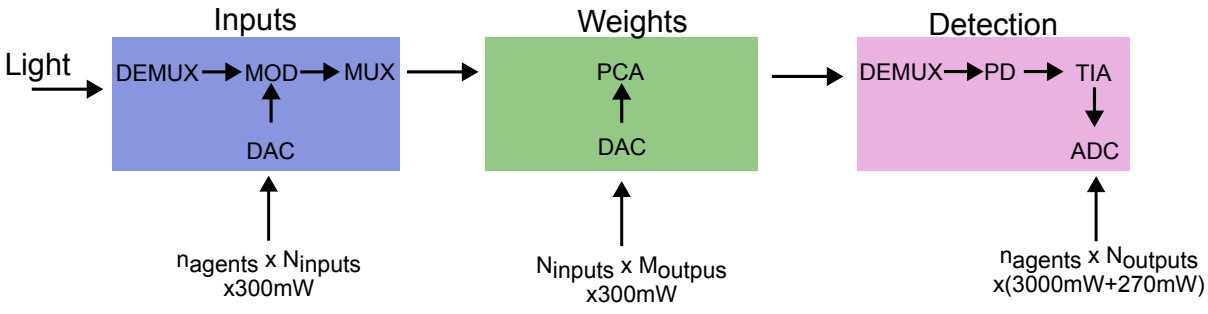


FIG. S6: **Photonic Crossbar Efficiency**. A sketch of the power consumption projection in different sections of the proposed system, based on parameters from the present demonstrator.

Architecture:

Optimization problems solved in this work use iterative algorithms. In such algorithms, the goal is to repeatedly update a system state until convergence is reached, meaning each iteration depends on the result of the previous one. In the photonic crossbar architecture considered here, each iteration consists of a MVM followed by feedback that updates the input vector for the next cycle. Because the computation is inherently sequential, the next iteration cannot begin until the previous round-trip through the photonic and electronic components is completed. As a result, the dominant performance metric is the latency per iteration, i.e., the time required for a full compute-feedback loop. Therefore, the relevant performance metric is the *round-trip latency per iteration*, rather than raw throughput of the analog computing system.

We model the iteration time as the sum of four contributions: optical propagation through the photonic crossbar, optoelectronic conversion and modulation, digital interconnect and logic delay, and a periodic digital batch-processing step. The baseline iteration time is therefore written as $t_{\text{base}} = \alpha t_{\text{opt}} + t_{\text{comp}} + t_{\text{buffer}} + t_{\text{logic+interconnects}}$. The optical propagation time across the array is given by $t_{\text{opt}} = \frac{2N \cdot \text{pitch} \cdot n_g}{c}$, where N is the array dimension, pitch is the waveguide spacing, n_g is the group index, and c is the speed of light. α represents a scalar multiplier accounting for optical delay from other parts of the photonic circuit, and is approximated with a constant value of 3. The optoelectronic response and buffering delays are estimated from the modulation bandwidth BW, such that $t_{\text{comp}} = t_{\text{buffer}} = \frac{1}{BW}$. We find that under this framework, and when considering different architectures for data flow and computation, the value of $t_{\text{logic+interconnects}}$ becomes the dominant parameter. This leads to two distinct operating regimes. In the first regime, the iteration latency saturates to its minimum value. This occurs when the feedback logic delay falls below the optical transit time and the response limits of the photonic components (namely the modulators and photodiodes). In this case, the electronic circuitry is tightly integrated with the photonic chip using short interconnects, and the required operations are implemented with simple, dedicated logic (e.g., thresholding using XOR/MUX operations). Such functionality can be realized as an application-specific integrated circuit (ASIC) that co-integrates photonic and electronic elements. The second regime arises when data movement becomes longer, for example when signals must be transferred off-chip to an FPGA or another external processing unit, or when each iteration requires more general-purpose digital processing rather than fixed-function logic. In this case, the latency increases as both the interconnect distance and the computational overhead of the programmable processor grow. Given that combinatorial optimization problems are largely discrete, thresholding operations tend to dominate the computation. This allows the system to approach the latency plateau, which is ultimately limited by the throughput of optics.

To account for occasional digital operations in our dynamic referencing algorithm (e.g., DMA transfer and a digital matrix multiply for energy calculations), we introduce a periodic penalty t_{penalty} that occurs once every P iterations. Following Amdahl-type reasoning, the effective contribution of this step is proportional to its fractional occurrence $f = 1/P$. The average iteration latency is therefore modeled as $t_{\text{avg}} = t_{\text{base}} + f\, t_{\text{penalty}} = t_{\text{base}} + \frac{t_{\text{penalty}}}{P}$,indicating that the occasional digital operation is strongly diluted and does not significantly affect the overall iteration rate. In this work, we demonstrate a proof-of-principle implementation. Realizing a tightly integrated architecture is part of our planned future work.

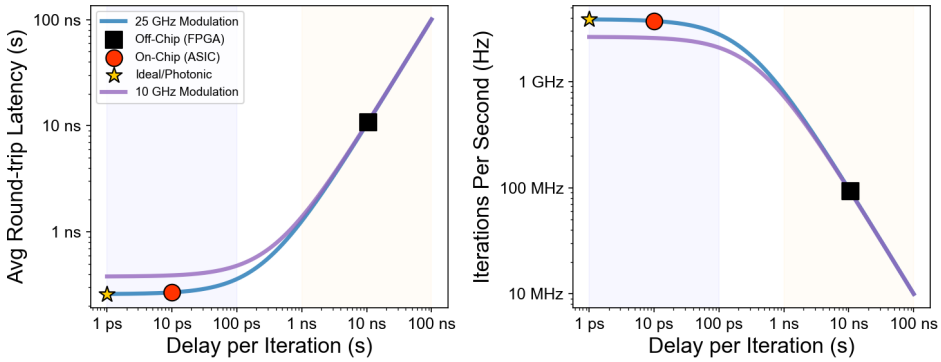


FIG. S7: **Latency and Throughput Projections**. Scaling of iteration latency and throughput for a 64×64 photonic array across different device bandwidths.(A) Total latency reveals a transition from a logic bound regime to a hardware (interconnect and compute complexity) bound plateau. (B) System throughput saturates as feedback logic delay drops below the optical transit time and device response limits.

SUPPLEMENTARY INFORMATION S5. COMPARATIVE ANALYSIS

In AI inference on GPUs and TPUs, the objective is to maximize throughput by aggregating independent requests into large batches, effectively trading increased per-request latency for higher computational efficiency via parallel processing of batch size. This is achieved with deep pipelining.

However, combinatorial solvers face a lack of possibility to do batching due to sequential data dependencies; the input to iteration $n+1$ depends on the output of iteration n, preventing the aggregation of independent requests. This is a sequential nature of data flow. Consequently, throughput is not a crucial metric; instead, the total time to solution is the sum of critical path delays ($T_{\text{total}} = \sum_{i=1}^{N} t_{\text{MVM}}^{(i)} + t_{\text{overhead}}^{(i)}$), a single MVM latency directly reduces the end-to-end latency.

This fundamental mismatch between the batch-oriented architecture of conventional GPUs/TPUs and the latency-sensitive nature of sequential optimization suggests that a fundamentally new hardware paradigm is required, and is the proposal of our work (arXiv:2511.00186 (2025)). Integrated photonic computing with photonic crossbars is particularly suited to fill this niche because light propagates through waveguides at speeds that introduce near-zero latency overhead per operation, and optical signals can perform analog MVMs in a single time step without the need for batched throughput to hide memory delays. In the table below, we compare photonic crossbar architectures with digital and emerging electronic in-memory computing architectures in terms of MVM performance metrics. In photonic architectures the MVMs can be enable sub-nanosecond MVM latency. However, to fully leverage these speed gains, the peripheral electronics that support auxiliary computations ($t_{\text{overhead}}^{(i)}$) must also operate with very low latency, on the order of a few cycles. This is another characteristic that makes combinatorial optimization problems a natural fit for the application. We cast the problems as arbitrary quadratic forms with binary vectors. It is therefore possible to implement an equivalent nonlinear threshold comparison with stored vectors using high speed CMOS comparators in the nanosecond time scales. The comparator outputs are then fed back recurrently as the input vector at step t+1 to the MVM unit.

Platform	**Size**	**Efficiency (TOPS/W)**	**Density (TOPS/mm^2)**	**Latency (ns)**
NVIDIA B200$^{\alpha}$	variable	9	5.5	N.A.
IBM Hermes$^{\beta}$	256×256	9.76	1.55	127 ns
Lightmatter$^{\gamma}$	128×128	0.81	0.047	1–10 ns
MRR weight banks$^{\delta}$	≤100×100	≤15	≤0.5	≤ 1
Crossbar arrays$^{\eta}$	≤100×100	≤0.3	≤2	≤ 1
MZI-Meshes$^{\zeta}$	≤800×800†	≤9	≤10	≤ 1

Table S1. Performance comparison of representative computing platforms. IBM's Hermes chip employs analog in-memory computing based on non-volatile phase-change memory devices, while Lightmatter's platform utilizes optical broadcast with electronic weighting and accumulation. The values shown for the three photonic computing platforms are extrapolated based on specific architectural assumptions and configurations. While photonic computing can compete in throughput at large matrix dimensions—as reflected in both compute throughput and compute density metrics—it particularly excels in achieving very low computational latency, the most critical metric for recurrent problems such as combinatorial optimizations. Ref: $^{\alpha}$ Shebanow, M. et al. Neurophos resources (2025), $^{\beta}$ Le Gallo, M. et al. Nat. Electron. 6 (2023), $^{\delta}$ Ahmed, S. R. et al. Nature 640, 368–374 (2025), $^{\eta}$ Tait, A. N. et al. Sci. Rep. 7, 1–10 (2017) and Blow, E. C. et al. Adv. Photonics Res. 5 (2024), $^{\zeta}$ Feldmann, J. et al. Nature 589, 52–58 (2021) and Frank, B. P. et al. doi.org/10.21203/rs.3.rs-7859321/v1 (2025).