Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Fully Parallel Multi-Agent Photonic Optimizer

Ghazi Sarwat Syed

ghs@zurich.ibm.com

IBM Research - Zurich https://orcid.org/0000-0002-2054-4887

Philipp Schmidt

Ruprecht-Karls-Univ Heidelberg

Frank Brückerhoff-Plückelmann

University of Heidelberg https://orcid.org/0009-0009-7713-6735

Jelle Dijkstra

Ruprecht-Karls-Univ Heidelberg

Wolfram Pernice

Heidelberg University https://orcid.org/0000-0003-4569-4213

Abu Sebastian

IBM Research Europe https://orcid.org/0000-0001-5603-5243

Article

Keywords: Optimization Problems, Nature Inspired Computing, Integrated Photonics

Posted Date: November 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10831844/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: There is **NO** Competing Interest.

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*

ABSTRACT

Optimization problems are central to many important cross-disciplinary applications. In their conventional implementations, the sequential nature of operations imposes strict limitations on the computational efficiency. Here, we discuss how analog optical computing can overcome this fundamental bottleneck. We propose a photonic optimizer unit, together with supporting algorithms that uses in-memory computation within a nature-inspired, multi-agent cooperative framework. The system performs a sequence of reconfigurable parallel matrix–vector operations, enabled by the high bandwidth and multiplexing capabilities inherent to photonic circuits. This approach provides a pathway toward fast paced and high-quality solutions for difficult optimization and search problems.

Keywords: Optimization Problems, Nature Inspired Computing, Integrated Photonics

INTRODUCTION

In the nature, many species engage in social associations for a variety of reasons. One of the most common being social foraging, which is the process by which individuals in a group work together to forage for resources (1–4). Such cooperation confers significant mutual benefits, including improvements in search efficiency and the dissipation of risk factors. Key to the success of this tactic lies predominantly in an improved exploration and exploitation balance that is achieved through the social and cognitive decision-making abilities of the interacting individuals (see Figure 1a). Several algorithms in computational science take inspiration from such "cooperative" behavior to solve difficult mathematical problems. Broadly, these include algorithms that implement cooperative artificial intelligence (5–10) and solve optimization problems. Multi-agent-reinforcement learning and metaheuristic algorithms such as ant colonies and particle swarms are some examples, which implement decentralized collaboration and coordination among synthetic individuals (agents) to accomplish their tasks. When applied to deep neural networks and *np* hard optimization problems, such algorithms have resulted in impressive results, such as emulating the game of soccer (5 and solving combinatorial (10) problems.

Nonetheless, such results continue to be based on the digital von Neumann computing systems, where data needs repeated shuttling between the processor and the memory units, on power-hungry interconnects (11). This presents a limit on the achievable compute latency and incurs notable energy costs. In addition to this inherently rooted complexity, such computing systems can allow only sequential implementation of cooperative algorithms. Crucially implying that each agent must be processed individually, and the combined data further processed. This results in the real-time interaction aspect and learning being lost. We also note that, in contrast to deeply pipelined workloads such as AI inference, where throughput is maximized by batching independent inputs, optimization problems exhibit an inherently sequential data flow. As a result, throughput is not the dominant performance metric. Rather, the total time to solution is governed by the accumulated per-iteration delays, such that reducing the latency of each iteration directly translates to lower end-to-end latency. Whilst various custom silicon computing hardware (i.e. ASICs and GPUs) have been developed to improve these aspects (12–15), they still depend on the same underlying electronic components, which are fundamentally limited by the electronic physics of Joule heating, RF crosstalk, capacitance, and by sequential bus-based data addressing architecture, which constraints data movement efficiency from limited bandwidths and cache coherency. For these reasons, there are strong incentives for developing processing-in-memory accelerators (16 and 17) and supporting algorithms that allow for rapid solving of difficult problems in a manner that is rigorous and nature-inspired on hardware (see Figure 1b).

Here, we propose just such a framework in the form of an integrated photonic optimizer (IPO), which is an iterative solver. We aim to solve continuous and discrete optimization problems using the proposed IPO. Central to our approach is the exploitation of physical properties of integrated photonic circuits and devices (18 and 19), along with the wavelength division multiplexing (WDM) capabilities of optics (21–24), to perform multiply-and-accumulate (MAC) operations efficiently. These computations are executed in-place and in parallel. By formulating the optimization problem as iterative analog MAC operations, we facilitate parallel exploration of the objective (cost) function landscape, targeting the sub-nanosecond latency achievable in optics (24). In this scheme, agents are uniquely encoded into non-overlapping wavelengths of light. Each wavelength independently explores

the solution space, shaped by the programmable configuration of the compute elements. This process occurs simultaneously across all agents and is guided by cooperative dynamics inspired by nature (see Figure 1b). Unlike traditional electronic architectures, where each agent is computed in separate hardware, our photonic system favors solutions that minimize information gain, thereby improving the chances of finding the global optimum. Moreover, by performing computation directly where data is stored, this approach bypasses the memory wall bottleneck and overcomes the speed limitations of electronic components.

RESULTS

Formal Definitions:

Computing optimizations are common computational operations in which the goal is to find the best solution from all feasible solutions with in the configurational search space. In formal terminology, an optimization problem can be defined in terms of a tuple $S, f, r)$. Here, S represents the search (state) space of the problem, r the set of the problem's constraints, and f is the objective function. The objective function guides the search process, such that $f : S \rightarrow \mathbb{R}^m$. In most problems, the directive is either to minimize or maximize the objective function, i.e. $\underset{\mathbf{s}\in S}{\text{minimize}} f(\mathbf{s})$ or $\underset{\mathbf{s}\in S}{\text{maximize}} f(\mathbf{s})$, and $\mathbf{s}$ is a vector, whose elements represent a decision variable that can be continuous or discrete. When the variables are discrete, the optimizations become discrete optimization problems. These are computationally harder to solve because an optimal solution has to be identified from a finite set of solutions. Integer and combinatorial optimization problems are examples of such problems, which are relevant in several technologically and scientifically important applications. The choice of an objective function is crucial in determining the quality of a solution, and while often given for integer type problems, must be carefully devised for the combinatorial problems. Canonical examples include Lyapunov functions, such as those associated with the Ising and Hopfield models.

When solving such problems, the naive approach is to randomly initialize a candidate query that performs search operations "locally" on the solution landscape. During this process, the solution evolves to provide an improvement to the solution until the optimal answer or a convergence criterion is met. In cases where the landscape profiles are complex, the solutions can easily get stuck in the local minima. A possible analogy can be drawn from the descriptions of foraging by an avian predator (see illustrated in Figure 1a). With a limited sensory range, through movements, a solitary bird scans a terrain while foraging. When the terrain is large and complex, chances are the bird may converge to no gains (improvements), or a sub-optimal food resource (local minima). Such predicaments are avoided when foraging in large flocks. In a flock, birds can discreetly examine different regions of the terrain, and through sensory communication choose to relay the information to the entire swarm. In effect, this scheme provides a means to "globally" search the landscape, thus, improving and speeding up the chances to locate resources.

In Figure 1a, we illustrate a hypothetical solution landscape to an objective function. Multiple attractor points provide local (sub-optimal) solutions. At any time instance, multiple disparate queries $\{\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^n\}$ in the state space are allowed to scan the landscape. Each query belongs to a discrete agent described by a tuple $(\mathbf{s}, \vec{d}, F)$, where F is the feedback given to an agent by the ensemble and $\vec{d}$ is its action. The exact implementation of these quantities depends on the chosen algorithm and we will elaborate on these terms in the following sections. It is suffice to suggest that an agent interacts with the environment and other agents in steps in time (t), to determine $\vec{d}^t = f(F)$. Such a scheme makes it possible that the agents, independently of their starting points, can converge to better solutions through dislodging from parasitic local minima. An IPO unit can realize this functionality (Figure 1c). Specifically, it enables a multi-agent solver in which multiple agents, mapped to independent input vectors, explore the same cost landscape through matrix–vector operations. A set of update rules, which are defined by curated algorithms guide the agents toward convergence to near-optimal solutions (see Figure 2a).

Conceptual Hardware:

We instantiate an optimization problem to use iterative MAC operations, between a $m \times n$ matrix that maps the objective function and the problem constraints, and the input vectors $\mathbf{s}^i$ that encode the state space. Figure 2b illustrates the proposed IPO unit. Here, each agent is assigned a distinct wavelength channel. The unit has a compute core which takes advantage of data stationarity by keeping MAC computation and data in the same place. This is achieved by storing the numeric values of a matrix directly in tunable attenuators (reconfigurable and analog memory cell). Potential realizations for these attenuators are for example electric absorption modulators (EAM) or phase change material (PCM) patches (22, 25, and 26). These memory cells are arranged at each intersection of a photonic crossbar circuit. Each memory cell performs a scalar multiplication operation of the form $v_{\text{out}} = w \times v_{\text{in}}$, where w is the transmission state of the cell and v_{in} is the input optical signal (22). The summation of the individual products occurs in the photodetector at the end of each waveguide. The input vector, representing an agent ($\vec{s}_i$), is encoded in the amplitude of the optical signals sent to the input waveguides of the crossbar. De-multiplexed input encoding

and detection, together with the large optical bandwidth allow all agents to be processed in parallel. Moreover, a recurrence exists in which the agents, updated from the output of the feedforward MAC operations, are fed back as inputs for the next iteration.

Importantly, wavelength-division multiplexing (WDM) enables multiple input vectors or agents to be encoded in the intensities of distinct, non-overlapping wavelength channels. These channels can propagate through the compute core simultaneously, allowing several multiplication operations to be performed in parallel. A large number of agents can be generated from broadband light sources such as amplified spontaneous emission (ASE) sources, frequency combs (27), or laser arrays. Photonic crossbars generally require each input waveguide to be mutually incoherent with all others to prevent interference at the detector. Consequently, the proposed IPO would require two cascaded demultiplexers and a total of $n_{\text{agents}} \cdot n_{\text{inputs}}$ non-overlapping spectral lines when coherent sources are used. Alternatively, inherently incoherent light, such as that provided by ASE sources, can be employed. In this case, the second demultiplexer can be replaced by a symmetric splitter, as illustrated in Figure 2b. After encoding, the channels are multiplexed into the crossbar and demultiplexed prior to readout by a photodetector (PD) array. While ASE light introduces significant inherent intensity noise, this can be mitigated to some extent by adjusting the optical bandwidth, optical power, or through averaging (28). However, randomness is an intrinsic feature of some optimization algorithms, and in such cases, we can harness this noise to our advantage.

Mathematically, at each iteration k, the agents in the IPO collectively perform the operation $\mathbf{s}_{k+1} = \mathbf{W}\mathbf{s}_k + \boldsymbol{\eta}$, where $\mathbf{s}_{k+1}$ denotes the recurrently updated state vector obtained from the previous state $\mathbf{s}_k$. Here, $\mathbf{W} = [w_{ij}]$ is the weight matrix, $\mathbf{s}_k = [s_k^1, s_k^2, \ldots, s_k^J]^T$ is the input state vector across agents at iteration k, and $\boldsymbol{\eta}$ represents additive noise. Crucially, all these operations are performed in parallel, eliminating also the need for explicit summation operations.

Experimental Validation:

So far we have discussed a generic IPO that provides a compact means of processing multiple agents with real-time cooperation within a single processing core. We now present a proof-of-concept experiment to evaluate the feasibility of the key functionalities required in the IPO. The setup (see Figure 3a-c) consists of an ASE source emitting broadband chaotic light over the 1530–1560 nm range. This light is spectrally filtered by a wavelength-division multiplexer into four 200 GHz bandwidth channels, centered on specific ITU C-Band wavelengths: C28 (1554.94 nm), C30 (1553.33 nm), C32 (1551.72 nm), and C34 (1550.12 nm). The filtered light from a single channel is then amplified in two stages and injected into a integrated photonic circuit (PIC) consisting of a 9×3 crossbar array containing electronic absorption modulators (EAM) at the cross point junctions. It also features EAMs on the input waveguides as well as integrated PDs at the output. The PD signal is amplified by off-chip TIAs and then fed to ADC ports of a FPGA board. The same board is also used to interface with the weight and input EAMs and thus constitutes the control unit of this experiment.

The optical noise arising from the ASE as well as the electrical noise injected by the receiver components lead to noisy MAC operations. We can reduce this accumulated noise via averaging multiple iterations of the same matrix-vector multiply (MVM) operation. We approximate the combined noise distribution as gaussian and thus are able to generalize the dependency of the noise on the amount of averages n_{avg} with the form, $f(n_{\text{avg}}) = \frac{A}{\sqrt{n_{\text{avg}}}} + B$, where A and B are fitting parameters that determine the initial noise amplitude and residual ground error, respectively. We benchmark the system by performing matrix-vector multiplications with random input vectors and constant matrix weights. The resulting output vector $\vec{y}$ is compared against the calculated ground truth using the L_2 error $\frac{\|\vec{y}_{\text{measured}} - \vec{y}_{\text{calculated}}\|_2}{\|\vec{y}_{\text{calculated}}\|_2}$. Figure 3d illustrates the averaged L_2 error values across the independent input channels, representing the injected noise that can be reconfigured through averaging. Note that this is one implementation scheme enabled by our current hardware resources. Controlled noise injection, either directly at the source or at the output, provides a more direct mechanism for tunability (21). In the following sections, we use the data obtained from these measurements to evaluate the feasibility of our proposed IPO unit.

Algorithmic Strategies on the IPO Platform:

In multi-agent optimization, each agent's behavior is influenced by a social interaction function F. During optimization, the system converges to a state that minimizes the net loss. Feedback can be applied either deterministically or via stochastic and evolutionary changes. When no feedback is present between agents ($F = 0$), the process is deterministic and corresponds to *sampling*. The manner in which feedback is implemented enables novel algorithmic advances. We introduce a general *dynamic referencing algorithm* (see Algorithm 1). The algorithm offers a candidate solution $\mathbf{s}^c \in S(t)$, defined as $\mathbf{s}^c = \arg\min_{\mathbf{s}\in S(t)} f(\mathbf{s})$ at time step t. The objective is to apply controlled random perturbations or operators to configurations other than $\mathbf{s}^c$, such that they differ minimally from $\mathbf{s}^c$ depending on the nature of F. These perturbations are designed to balance exploitation of the already

obtained knowledge by the agent ensemble and exploration of individual agents independent of the combined knowledge. They can change over time adjusting the this balance. As a convergence criterion we introduce a similarity metric between $\mathbf{s}^{\mathrm{c}}$ and the other configurations. For discrete inputs, this similarity can be, for example, the Hamming similarity $\mathrm{Sim}_{\mathrm{Hamming}}(\mathbf{s}^{\mathrm{c}},\mathbf{s}^{\mathrm{n}})$. This approach allows multiple input configurations to converge toward a single, high-confidence solution. There is a one-to-one correspondence between the number of agents (or wavelengths) and the solution quality: a greater number of agents enables broader exploration of the solution landscape, improving the chances of locating optimal regions and boosting convergence likelihood. Moreover, this aligns with the idea that confidence scales inversely with the Shannon entropy (29), defined as $E_{\mathrm{S}} = -p\log p-(1-p)\log(1-p)$ where $p=\frac{N_c}{N}, \quad 1-p=1-\frac{N_c}{N}$. Here, N is the total number of parallel queries at a given iteration, and N_c is the count of configurations matching the candidate solution $\mathbf{s}^{\mathrm{c}}$. As more configurations align with $\mathbf{s}^{\mathrm{c}}$, the entropy E_{S} decreases, indicating growing consensus and confidence in the solution.

Algorithm 1 Dynamic referencing algorithm

1: **(Best Selection)** Identify current best: $\mathbf{s}^{\mathrm{c}} \leftarrow \arg\min_i E_i$
2: **(State Adaptation)** Apply every k iterations:
3: For each agent $\mathbf{s}^{\mathrm{i}} \neq \mathbf{s}^{\mathrm{c}}$:
4: **Update:** apply deterministic Update
5: **(Optional Enhancements)** Apply as needed:
6: **Crossover:** exchange information with another agent
7: **Annealing:** accept probabilistic pertubation
8: **Elitism:** exclude top agents from update
9: **(Energy Update)** Recompute $E_i \leftarrow f(\mathbf{s}^{\mathrm{i}})$, refresh $\mathbf{s}^{\mathrm{c}}$

We now discuss some schemes to implement the perturbations. The first is the case of photonic, multi-agent optimization analogous to simulated annealing (8 and 30. In this approach, injection of uncorrelated fluctuations on either the state itself or the deterministic update rule, enabled by a reconfigurable η can dislodge agents into probable state space from parasitic minima. A key requirement for this method is high-bandwidth random number generation and a decay profile applied to η which, in our implementation, is naturally provided by the analog noise inherent to the IPO hardware. We combine this with the evolutionary principle (31 and 32) of crossover ($\bigoplus_f$) where in each cycle the agent inherits part of the best current solution. Our second approach implements particle swarm optimization (PSO), where agents explore the loss landscape based on physical principles of velocity and interaction, updating their trajectories based on their own and the ensemble experience (33 and 34). In the following section, we emulate these algorithmic features under realistic conditions. Although the examples we present are not exhaustive, they are meant to illustrate and benchmark the unique capabilities enabled by the IPO unit.

Solving Discreet Optimization:

As a first example, we discuss the case of discrete optimization problems. We solve the constrained optimization graph problem of Max-cut (35 and 36. Max-cut (see Figure 4a) is typically utilized in integrated-circuit designing and imaging, and the goal is to partition a graph's vertices into two complementary sets (highlighted in different colors), such that the number of edges linking the two sets is as large as possible. Stated formally, given an undirected graph $G(v,e)$, where v and e are the graph's vertices (nodes) and the edges (interconnections) between nodes, respectively. The goal is to find a subset $s \subseteq v$, such that the number of edges between s and its complement subset is maximized.

An objective (energy) function f must use the MAC operation in converging to a solution. We use the framework of a *Hopfield model* to achieve this (37 and 38). In this scheme, we map the the graph problem to the energy function $f=-\frac{1}{2}\sum_{\mathrm{mn}}^{\mathrm{N}} -w_{\mathrm{mn}}v_{\mathrm{m}}v_{\mathrm{n}}$ where w_{mn} is the adjacency matrix that encodes the strength of the edges. v_{m} and $v_{\mathrm{n}} \in \{-1,1\}$ represent the state of interconnected nodes in a network of N nodes. The energy has an arbitrary unit scale, which is a function of the graph problem (connection density and weight values), and the values of the binary nodal states denoted as H and L. To find an optimal solution, the energy function is iteratively minimized by toggling the nodal states. This is done by using the update-rules: $v_m = \begin{cases} H & \text{if } \frac{\sum_{m\neq n} w_{\mathrm{mn}}v_{\mathrm{m}}}{\gamma}+\eta \geq \theta_{\mathrm{ref}} \\ L & \text{if } \frac{\sum_{m\neq n} w_{\mathrm{mn}}v_{\mathrm{m}}}{\gamma}+\eta < \theta_{\mathrm{ref}} \end{cases}$ for some threshold θ_{ref} and some noise value η drawn from a normal distribution. The normalization constant $\gamma = \max_{v\in G}(\mathrm{D})$ equals the maximum degree D of all graph vertices. This ensures the MAC result to stay in the interval [-1, 1]. When the optimization is performed, the system converges to the nodal state distributions that minimize the net energy. Each agent optimizes independently by picking random nodes and applying the update rules. In regular intervals the agents interact via crossover and each agent inherits a number of nodal states from the current best agent. The number of nodes inherited increases with time. The specific schedule is elaborated on in the methods section. The algorithm is outline in

pseudocode as Algorithm 2.

As an example, we solve a cyclic graph of 40 nodes. Briefly, the system is initialized with a random distribution of the nodal states, where some nodes belong to a set Set_1, and others to the complementary set Set_2. Each node can be in a state of H or L. Furthermore, each node has edges with only immediate neighbors ($w_{\mathrm{mn}} = w_{\mathrm{nm}}$), and there are no self-loops ($w_{\mathrm{mn}} = 0$, if $m = n$). Four random solution candidates representing individual agents (wavelengths) are initialized. We choose 0 as the threshold value θ_{ref}. Although the multi-agent approach is capable of optimal convergence, this typically necessitates an increase in both the number of agents and computational iterations. Combining them with photonic annealing, however, improves the convergence success rate.

We emulate different noise decay profiles and benchmark the performance against the experimentally obtained noise profiles and fully deterministic optimization (see Figure 4b-c). See methods for the rescaling method of the noise benchmark data. The injected noise profiles follow the form earlier described with $B = 0$ and $A = \sqrt{\mathrm{cycles}/\alpha}$, where we sweep the α parameter to obtain the different profiles, as shown in Figure 4c. These values dictate the standard deviation of the normal distribution used to draw the values η in the update rules. Figure 4b shows the individual Hopfield energies of one example simulation using the experimental noise profiles. Each noise profile is executed 20 times and we record the average Hopfield global energy in Figure 4c. The optimal solution for this problem has an energy of -40. When noise is injected the solver reliably finds the correct solution regardless of noise magnitude within the emulated noise range. The chosen noise profile seem to affect convergence speed, with higher initial noise profiles delaying discovery of the optimal solution. However, profiles with low initial noise below ≈ 0.5 show very similar behavior to each other. This suggests that the system is robust against high initial noise when agents are more in the exploratory phase, but indifferent to noise below some threshold. We also observe that the noise profiles extrapolated from data are among the best in terms of convergence speed. Thus, we expect the proposed IPO system to perform well on this problem based on these simulations.

To benchmark our approach we solve 30 non-circular graphs from the open sourced BiqMac-Library (39). We use 4 agents and the experimentally obtained noise profiles. These problems cover undirected and unweighted graphs with edge probability of 50% of sizes 60, 80 and 100 nodes with 10 examples each. Each graph is solved multiple times, and the solutions are tabulated to compare the success rates. Already without any hyperparameter tuning, we are able to find the most optimal solutions of every graph (see bottom panel of Figure 4d), at least more than once. This is represented as success rates, which vary between 10% and 100% showing slightly worse rates for the graphs of large sizes. Despite the low success rate for some graphs, we note that many obtained solutions are within 1% of the true solution, even when the algorithm does not fully converge. This is shown in the top panel of Figure 4d where we plot the distribution of the relative remaining error for all runs. We note that the success rates could be favorably improved by increasing the number of agents and further optimizing the noise profiles and nature of the inter-agent cooperation.

Solving Continuous Optimization:

We now discuss the utility of the IPO in linear programming, i.e., solving optimization problems with linear objective functions and linear constraints. We discuss the solution to a constrained optimization problem. These are problems in which a function is to be minimized or maximized, subject to certain constraints. These problems are relevant in a range of applications, for example, in manufacturing where production is more generally constrained by resources and demands. In Figure 5a we illustrate the mapping of a constrained optimization problem onto the compute core. Two equations describe the objective function and the constraint. To achieve parallel computations on both equations, we map the coefficients of the objective function onto the transmission states of the memory cells at the cross-points of one column and the constraint function to a second column. We use PSO as the optimizer for this problem. In PSO, a collection of agents move throughout a region in discrete steps. After each iteration, the algorithm updates the velocity $\vec{v}$ of all agents, nudging them towards the best solution (33, 34, and 40. We elaborate on the algorithmic specifics in the methods section and Algorithm 3.

In our proposed IPO both the loss and constraint evaluation are performed in-memory in the photonic crossbar array for all agents/particles in a single compute cycle. As an example, we evaluate the minimum of the constrained linear function, $f(x_1, x_2) = 10x_1 + 20x_2$, while $3x_1 + 4x_2 \geq 60$ and $x_1 - 2x_2 \leq 0$. We normalize the loss landscape to the intervall [-1, 1], where -1 is the minimum within the constrained region and +1 the maximum based on the bound on the initial positions. We also constrain the area based on minimal and maximal values of the coordinates $x_1 \in [-27.5, 27.5]$ and $x_2 \in [-5, 50]$. Figure 5b shows particle trajectories through the constrained landscape for a swarm size of 6 particles. Note, that all agents explore the solution landscape, constantly referencing other agents and adjusting their velocity based on their own experience and those of other agents. The constraints are implemented as absorptive walls meaning agents hitting a constraint are set to the boundary with their velocity set to zero (41 and 42).

We benchmark this implementation against increasing uncertainty in the evaluation of the loss function as well as the constraints. Given the continuous nature of the variables, the compute imprecision can be more deteriorating in linear programming compared to the discrete case where after performing a MVM, the outputs can be binarized by thresholding to produce discrete variable values. Here, noise affects optimization as it induces uncertainty in the quality of each agent's present and past positions. This is illustrated in Figure 5d, where we plot the global minimum swarm loss over time of an example simulation using 10 particles. We see that the perceived minimum loss, i.e. the minimum loss the swarm recorded is lower than the true loss at the recorded positions. Thus, the estimates of the current best position are likely often slightly incorrect, pulling the swarm in a slightly wrong direction.

In addition, noisy constraint evaluation can lead to agentic positions being incorrectly accepted or rejected. However, owing to its low-latency computational nature, the approach can already provide a high-quality starting point (warm-up) for initialization, offering an advantage over purely random initialization (47). To still obtain solutions within the bounds we apply a single noiseless boundary condition check on the personal best positions of all agents within the swarm, rejecting out of bounds agents at the end of each optimization. The distance to the optimum to the thus obtained best legal minimum is plotted in figure 5c. We repeat each configuration for 10 different random initial particle positions and velocities and take the average over all runs that produced at least one minimum within bounds. We find that noise can be readily compensated by increasing the swarm size. Based on the minimal noise obtained from our benchmark experiment we expect our IPO to perform well on this problem with agent sizes larger than 5. As a reference we also mark noise values equivalent to $1/2 \cdot 1/2 \cdot 1/2^b$ where b is a certain bit resolution $b \in \{4,6,8\}$. The 2σ interval of this standard deviation value equals half the level spacing of the respective bit-resolution. Thus, this value corresponds to a noise level equivalent to a $\approx 5\%$ bit-error probability on the least significant bit during discretization. Regardless, optimization problems needing high compute precision can benefit from stabilized light sources, such as mode-locked or narrowband lasers.

DISCUSSION:

Integrated photonic computing architectures can achieve significantly higher compute densities than digital electronics, owing to the speed of optics and the parallelism enabled by spatial multiplexing. When combined with non-volatile elements such as phase-change photonic memory devices, this approach can further facilitate fast, and efficient in-memory computing computational hardware (in supplementary information S1 we discuss IPO with PCM devices). Several optical architectures have also been demonstrated in the past few years for solving optimization problems. However, these are largely based on coherent networks that use degenerate optical parametric oscillators (43), injection-locked laser networks (44) or spatial multiplexing (45). Working with time-multiplexed optical signals , these, therefore, do not capitalize on the parallelism property, and face scaling limitations due to dispersion and decoherence. More recently, integrated Mach–Zehnder interferometers were demonstrated for coherent networks. Such on-chip networks significantly reduce the footprint, however, they are volatile and require high standby power. The latter also set a limit on the size of such networks, since individual driver units must support individual interferometers (in supplementary information S4, we compare these various different on-chip architectures).

In the ideal IPO, multiple MAC operations can be executed within sub-nanosecond latency, enabled by binary and pulse-amplitude input encoding (see supplementary information S4). The maximum achievable speed is ultimately limited by the speed of the modulators, photo-detectors and the bandwidths of the ADCs and DACs of the control board. Achieving comparable performance using conventional digital electronics remains extremely challenging. However, the estimation of the energy function and the corresponding feedback loop remain bottlenecks, as these steps are still executed electronically in each iteration. That said, this limitation is still less critical than in photonic neural network computing. Since our approach employs only a single layer and simple auxiliary operations, a high-bandwidth controller is sufficient to maintain the system in a compute-bound regime. Furthermore, because the the optical hardware remains idle during electronic processing, we can enable static batching, where the same wavelengths are reused across time slots to represent different agents. If P wavelengths are available and B agents are time-multiplexed, the total number of agents scales as $N_{\text{agents}} = P \times B$. Since photonic latency is inherently low, increasing B adds minimal overhead while significantly expanding parallel exploration. Nonetheless, an ideal IPO would have simpler update rules, which could be realized entirely in the optical domain. Realizing IPO in fully integrated photonic hardware may also present certain challenges, one of which is input encoding. This limitation can be addressed by using broadband incoherent light sources, thereby removing the need for cascaded de-multiplexers. Moreover, this approach can leverage the inherent entropy source (28), enabling random permutations to be incorporated into state updates. An IPO can also be envisioned with provisions for both deterministic and stochastic sources, or other noise injection mechanisms such as on the output nodes, allowing optimization problems that require high compute precision to also be solved with higher success rates. We also point out that the numeric precision of compute depends on whether the optimization problem is discrete or continuous. For example, in discrete optimization problems small errors in the MVM can be tolerated as long as the sign (threshold crossing)

is correct, making the operation robust to limited bit-width.

In our proposed method of solving optimization problems, we take a population-based meta-heuristic approach. We implement in-memory search operations by leveraging the parallelism from WDM in conjunction with utilizing inherent hardware noise to our advantage. While the forward-pass operation relies on MAC operations, similar to neural network implementations on photonic circuits, we demonstrate that this hardware can function as an iterative solver for optimization tasks. Specifically, we treat the photonic substrate as a single-layer compute unit, enhanced with recurrent connections, and for discrete optimization problems, the data flow is reduced to a binary representation (46). We also outline important algorithmic directions for implementing optimization problems. The use of multiple agents through dynamic referencing introduces an additional degree of freedom, enabling standard optimization algorithms to be realized more rigorously. Within this framework, information gain and entropy serve as principled metrics, which provide robust termination criteria. Looking forward, unique schemes such as annealing can be distributed across agents and coordinated through feedback mechanisms, thereby shaping a collective annealing schedule. Similarly, feedback-driven updates can be employed to guide solutions according to evolutionary principles. Here the optimizer can track the evolutionary progress of individual agents as well as subsets of populations in real time, closely mirroring how these processes unfold in nature (see Supplementary Section S1). Finally, while we demonstrate in preliminary simulations the end-to-end optimization of our proposed IPO unit it can also be of use in cases where it cannot solve the problem on it's own. The strong approximation capability for a good a solution can also enable the optimizer a valuable tool for initializing solutions (high-quality starting point compared to random initialization) in downstream annealing-based solvers (47 and 48).

In summary, we have presented a co-designed framework that combines photonic hardware with algorithmic strategies to tackle challenging optimization problems. We introduce an integrated photonic optimizer that leverages low-cost in-memory multiply-and-accumulate operations, implemented within a nature-inspired multi-agent cooperative scheme supported by multiplexing. As demonstrative examples, we use experimentally benchmarked hardware to emulate solutions for both continuous and discrete graph problems. By performing computation where the analog data resides and enabling truly parallel multi-agent processing, this approach overcomes the energy and latency limitations inherent to electronic systems. Furthermore, the approach exploits intrinsic hardware noise, transforming it into a computational resource of optimization problems.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[2] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[3] D. S. Schloesser, D. Hollenbeck, and C. T. Kello, "Individual and collective foraging in autonomous search agents with human intervention," *Scientific Reports*, vol. 11, no. 1, p. 8492, 2021.

[4] C. J. Torney, A. Berdahl, and I. D. Couzin, "Signalling and the evolution of cooperative foraging in dynamic environments," *PLoS computational biology*, vol. 7, no. 9, p. e1002194, 2011.

[5] A. Dafoe, Y. Bachrach, G. Hadfield, E. Horvitz, K. Larson, and T. Graepel, "Cooperative ai: machines must learn to find common ground," *Nature*, vol. 593, no. 7857, pp. 33–36, 2021.

[6] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pp. 66–83, Springer, 2017.

[7] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[8] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[9] A. Gogna and A. Tayal, "Metaheuristics: review and application," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 503–526, 2013.

[10] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

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

[12] Y. Zhou and Y. Tan, "Gpu-based parallel particle swarm optimization," in *2009 IEEE Congress on Evolutionary Computation*, pp. 1493–1500, IEEE, 2009.

[13] I. Chakroun, N. Melab, M. Mezmaz, and D. Tuyttens, "Combining multi-core and gpu computing for solving combinatorial optimization problems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 12, pp. 1563–1577, 2013.

[14] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, "An accelerator architecture for combinatorial optimization problems," *Fujitsu Sci. Tech. J*, vol. 53, no. 5, pp. 8–13, 2017.

[15] F. Zhang, C. Zhao, S. Han, F. Ma, and D. Xiang, "Gpu-based parallel implementation of vlbi correlator for deep space exploration system," *Remote Sensing*, vol. 13, no. 6, p. 1226, 2021.

[16] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*, pp. 171–243, Springer, 2022.

[17] A. Sebastian, M. Le Gallo, and E. Eleftheriou, "Computational phase-change memory: Beyond von neumann computing," *Journal of Physics D: Applied Physics*, vol. 52, no. 44, p. 443002, 2019.

[18] C. Ríos, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. D. Wright, H. Bhaskaran, and W. H. Pernice, "Integrated all-photonic non-volatile multi-level memory," *Nature photonics*, vol. 9, no. 11, pp. 725–732, 2015.

[19]M. Wuttig, H. Bhaskaran, and T. Taubner, "Phase-change materials for non-volatile photonic applications," *Nature photonics*, vol. 11, no. 8, pp. 465–476, 2017.
[20]S. Hua, E. Divita, S. Yu, B. Peng, C. Roques-Carmes, Z. Su, Z. Chen, Y. Bai, J. Zou, Y. Zhu, et al., "An integrated large-scale photonic accelerator with ultralow latency," *Nature*, vol. 640, no. 8058, pp. 361–367, 2025.
[21]C. Wu, X. Yang, H. Yu, R. Peng, I. Takeuchi, Y. Chen, and M. Li, "Harnessing optoelectronic noises in a photonic generative network," *Science Advances*, vol. 8, no. 3, p. eabm2956, 2022.
[22]J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, *et al.*, "Parallel convolutional processing using an integrated photonic tensor core," *Nature*, vol. 589, no. 7840, pp. 52–58, 2021.
[23]S. Ghazi Sarwat, F. Brückerhoff-Plückelmann, S. G.-C. Carrillo, E. Gemo, J. Feldmann, H. Bhaskaran, C. D. Wright, W. H. Pernice, and A. Sebastian, "An integrated photonics engine for unsupervised correlation detection," *Science Advances*, vol. 8, no. 22, p. eabn3243, 2022.
[24]S. Hua, E. Divita, S. Yu, B. Peng, C. Roques-Carmes, Z. Su, Z. Chen, Y. Bai, J. Zou, Y. Zhu, *et al.*, "An integrated large-scale photonic accelerator with ultralow latency," *Nature*, vol. 640, no. 8058, pp. 361–367, 2025.
[25]B. Dong, F. Brückerhoff-Plückelmann, L. Meyer, J. Dijkstra, I. Bente, D. Wendland, A. Varri, S. Aggarwal, N. Farmakidis, M. Wang, *et al.*, "Partial coherence enhances parallelized photonic computing," *Nature*, vol. 632, no. 8023, pp. 55–62, 2024.
[26]X. Li, N. Youngblood, C. Ríos, Z. Cheng, C. D. Wright, W. H. Pernice, and H. Bhaskaran, "Fast and reliable storage using a 5 bit, nonvolatile photonic memory cell," *Optica*, vol. 6, no. 1, pp. 1–6, 2019.
[27]T. Fortier and E. Baumann, "20 years of developments in optical frequency comb technology and applications," *Communications Physics*, vol. 2, p. 153, Dec. 2019. Publisher: Nature Publishing Group.
[28]F. Brückerhoff-Plückelmann, H. Borras, B. Klein, A. Varri, M. Becker, J. Dijkstra, M. Brückerhoff, C. D. Wright, M. Salinga, H. Bhaskaran, B. Risse, H. Fröning, and W. Pernice, "Probabilistic photonic computing with chaotic light," *Nature Communications*, vol. 15, p. 10445, Dec. 2024. Publisher: Nature Publishing Group.
[29]C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
[30]P. J. Van Laarhoven, E. H. Aarts, P. J. van Laarhoven, and E. H. Aarts, *Simulated annealing*. Springer, 1987.
[31]H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution algorithms in combinatorial optimization," *Parallel computing*, vol. 7, no. 1, pp. 65–85, 1988.
[32]S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2021.
[33]Y. Shi *et al.*, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, vol. 1, pp. 81–86, IEEE, 2001.
[34]D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft computing*, vol. 22, pp. 387–408, 2018.
[35]J.-Y. Potvin and K. A. Smith, "Artificial neural networks for combinatorial optimization," in *Handbook of metaheuristics*, pp. 429–455, Springer, 2003.
[36]B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*, vol. 1. Springer, 2011.
[37]J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons.," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.
[38]F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, *et al.*, "Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks," *Nature Electronics*, vol. 3, no. 7, pp. 409–418, 2020.
[39]A. Wiegele, "Biq Mac Library - BInary Quadratic and MAx Cut Library," 2007.
[40]M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements," in *2013 IEEE Congress on Evolutionary Computation*, pp. 2337–2344, June 2013. ISSN: 1941-0026.
[41]J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *IEEE Transactions on Antennas and Propagation*, vol. 52, pp. 397–407, Feb. 2004.
[42]T. Huang and A. Mohan, "A hybrid boundary condition for robust particle swarm optimization," *IEEE Antennas and Wireless Propagation Letters*, vol. 4, pp. 112–117, 2005.
[43]P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, *et al.*, "A fully programmable 100-spin coherent ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.
[44]S. Utsunomiya, N. Namekata, K. Takata, D. Akamatsu, S. Inoue, and Y. Yamamoto, "Binary phase oscillation of two mutually coupled semiconductor lasers," *Optics express*, vol. 23, no. 5, pp. 6029–6040, 2015.
[45]D. Pierangeli, G. Marcucci, and C. Conti, "Large-scale photonic ising machine by spatial light modulation," *Physical review letters*, vol. 122, no. 21, p. 213902, 2019.
[46]G. S. Syed and A. Sebastian, "Solving optimization problems with photonic crossbars," Sept. 23 2025. *US* Patent 12,422,882.
[47]M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.
[48]D. J. Egger, J. Mareček, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, 2021.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at....

Section S1. Phase Change Memory based Weights.
Section S2. Further discussion on Linear Programming Implementation.
Section S3. Precision of Compute.
Section S4. Architecture Analysis.
Fig. S1. Combinatorial Optimization Problems.
Fig. S2. Linear Programming Problems.
Fig. S3. Success rate and stability of the PSO solutions.
Fig. S4. Iterative Solver.

Fig. S5. Photonic Crossbar Efficiency.
Fig. S6. Latency and Throughput Projections.
Table. S1. Performance comparison of representative computing platforms.

ACKNOWLEDGMENTS

S.G.S and W.P acknowledge NEQIOS project funded by the European Union's Horizon Europe (grant agreement no. 101259183). S.G.S acknowledges partial funding from the European Research Council (grant agreement no. 101222715). A. Sebastian acknowledges partial funding from the European Research Council (grant agreement no. 101201020). W.P. acknowledges partial funding from the European Research Council (grant agreement no. 101200429). W.P. also acknowledges partial funding from DARPA NAPSAC program.

COMPETING INTERESTS

The authors declare no competing financial interests.

DATA, CODE, AND MATERIALS

This study did not generate new materials. The data that support the findings of this study are uploaded to a public repository (10.5281/zenodo.19660492).

CONTRIBUTIONS

G.S.S. conceived the research question and designed the algorithms and simulations. P.S. performed the hardware experiments and conducted the emulation studies. F.B.P. designed the experimental setup, which was built by J.D, and performed the initial hardware experiments. W.P. and A.S.E. provided technical guidance and management support. G.S.S wrote the manuscript with inputs from P.S. All authors read the manuscript.

METHODS

a. Scaling experimental Noise : Our performance benchmark experiment consists of 1280 random MVMs of the type $\vec{y} = M\vec{x}$, $\vec{x} \in [-1,1]\mathbb{R}^{64}, M \in [1]\mathbb{N}^{64\times 1}$. Each vector component is drawn from a uniform distribution. This operation is then averaged to reduce hardware noise stemming from the ASE source and the readout electronics, in order to obtain the noise profiles shown in Figure 3. Supplementary information Figure S5a shows the resulting scatter plot of the target MVM operation versus the measured MVM result for two different averaging factors. As indicated in the figure, the MVM values lie approximately within the interval $[-15,15]$. To obtain realistic noise magnitude estimates for different algorithms—where the results of the operations can vary significantly—we rescale these results. From the distribution of target operations, we choose the interval $[-3\sigma,3\sigma]$ to correspond to $[-1,1]$. From these normalized values, we obtain error distributions of Gaussian shape. These distributions narrow with increased averaging and are shown as heatmaps for all four measured channels in Figure S5b. We fit these distributions to extract the standard deviations for each averaging factor and then fit a $A/\sqrt{n}+B$ profile to the resulting values (see Figure S5c). The data reveal that the error distributions exhibit a slight deviation from zero mean, suggesting the presence of residual systematic error. We do not account for this offset when simulating our noise levels, assuming it can be mitigated through improved calibration in future experiments. As mentioned in the main text, all values with added noise are normalized to the same range of $[-1,1]$, allowing us to directly transfer our normalized noise values into the simulations.

b. max-cut solver The Max-Cut solver is initialized by assigning each agent a random cut of half the graph size. Each agent then optimizes independently by picking a random node and updates for 10 iterations. This concludes one optimization cycle after which the agents share solutions by inheriting a number of node configurations from the currently best performing agent, based on its Hopfield energy. The number of inherited nodes increases with cycles. The case of the circular graph we use 2000 cycles and share no nodes for the first 50 cycles, 25% of the nodes from cycle 50 to 100 and 50% of the nodes for all remaining cycles. Note that the shared nodes are chosen randomly, hence we are not necessarily sharing connected subgraphs. For the non-circular graphs, we also use 2000 cycles of length 10 and increase the cooperation every 400 cycles in 20% steps starting with 0% of nodes shared and ending at 80% nodes shared. For the emulated noise profiles all agents share the same decay traces, and the four measured channels are distributed to the four agents.

Algorithm 2 Photonic Max-Cut Solver

Require: Graph $G(V,E)$, objective function $f_{Hop} : \mathbf{s}^n \to \mathbb{R}$, number of agents n, noise profile $\eta(t) = \mathcal{N}(\mu, \sigma_t^2)$, number of cycles n, cycle length m

Initialize each agent's solution $\vec{s}^i \in \mathbf{s}^n$ and compute initial energy $E_i \leftarrow f_{\text{Hop}}(\vec{s}^i)$
for n cycles **do**
  for k iterations **do**
    (Energy Computation) All agents compute simultaneously: agent i
    pick random node g_i
    $g_i' \leftarrow \text{MAC}(\vec{s}^i) + \eta(t)$ ▷ Update node g_i
    $s^{i'} \leftarrow \text{UpdateState}(\vec{s}^i, g_i')$ ▷ Update Graph State
    $E_i' \leftarrow f_{\text{Hop}}(\vec{s}^i, g_i')$
  end for
  (Communication)
  $\vec{s}^c \leftarrow \arg\min_{\vec{s} \in S^n} f_{\text{Hop}}(\vec{s})$
  Increase $\leftarrow$ Similarity$(\vec{s}^i, \vec{s}^c)$
end for
Determine the best state among all agents
return best solution

c. cont. optimization We use canonical PSO with absorptive bounds as described in references (41 and 42) to implement particle swarm optimization. We use an adaptive random topology of size 3, meaning each particle informs at most 3 other particles beyond itself and shuffles these whenever a cycle did not improved the global optimum (40). The algorithm is outlined in references (34 and 40), and we briefly discuss it here. The velocity update equation for an agent $\vec{s}_i$ at iteration $t+1$, follows $\vec{v}_i^{t+1} = \omega\vec{v}_i^t + c_1\vec{u}_1^t \odot (\vec{p}_i^t - \vec{s}_i^t) + c_2\vec{u}_2^t \odot (\vec{l}_i^t - \vec{s}_i^t)$, where $\vec{p}^t$ is the current personal best position encountered by the particle and $\vec{l}^t$ the current local best within it's neighborhood. $\vec{u}_1^t$ and $\vec{u}_2^t$ are independent random vector drawn from a uniform distribution between [0, 1]. Note that $\odot$ is used to mark element wise multiplication. The hyperparameter constants are set to $\omega = 1/(2ln(2))$ and $c_1 = c_2 = 0.5 + ln(2)$. These are directly taken from reference (40). As mentioned before, bounds are implemented as absorptive walls, i.e. a particle about to break a constraint is set to the intersection of it's velocity vector and the boundary with velocity 0 (41 and 42). For a legal new position the particle is updated by simply adding the new velocity to the old position: $\vec{s}_i^{t+1} = \vec{s}_i^t + \vec{v}_i^{t+1}$.

We normalize the loss function to be -1 at the minimum and +1 at the maximum within the allowed initialization and computational bounds. For our example case we set the initialization intervals to $x_1 \in [-27.5, 27.5]$ and $x_2 \in [-5, 50]$. We sample random points in these intervals, and resample for any coordinates that break any constraint until all particles are initialized at a legal position. We let each swarm develop for 300 iterations. The bound values are normalized to the interval $[-1, 1]$ within this initialization area. To emulate noisy MAC-operations we draw a sample from a standard normal distribution scaled by the specific noise value of the run for both constraint and loss function evaluation. Due to noisy constraint evaluation it is possible that we obtain positions outside of the constraints. Thus, to still obtain a valid final result we we sample the best recorded position of each particle and perform one noiseless constraint evaluation, rejecting out of bounds positions and returning the best legal position as the final result. This can lead to rejection of of all minima turning the run unsuccessful. Supplementary Section S2 shows the number of successful runs across the noise vs. swarm size sweep shown in Figure 5.

Algorithm 3 Parallel Wavelength Exploration (PSO)

Require: Objective function $f : \mathbf{s} \to \mathbb{R}^m$, constraints r, number of agents N, neighborhood size K, number of iterations n

Randomly initialize each agent's solution $\vec{s}^i \in S$ and compute $f_i = f(\vec{s}^i)$, randomly select K other agents to inform (duplicates allowed).
for k iterations **do**
  (Parallel compute) All agents operate simultaneously:
  get personal best $\vec{p}^i$
  get local best $\vec{l}^i$ from informants
  $\vec{s}^{i'} \leftarrow \text{Update}(\vec{s}^i, \vec{p}^i, \vec{l}^i)$
  if $s^{i'}$ satisfies r **then**
    $\vec{s}^i, f_i \leftarrow \vec{s}^{i'}, f(\vec{s}^{i'})$
  end if
  compute new global optimum $\vec{g}'$
  if $\vec{g}' = \vec{g}$ **then**
    reshuffle neighborhoods
  end if
end for
return $\vec{s}^j \mid f(\vec{s}^j) = \min_{\vec{s}^i \in Swarm} f(\vec{s}^i)$

d. Experimental setup : The ASE lightsource (Agilent 83438A) is spectrally filtered by a wavelength division multiplexer and subsequently amplified by two amplifier stages (Pritel LNHPFA-30 pre-Amp into Pritel FA-33-IO-1807-23-009 power Amp) to approx. 100 mW

average power. The core of the IPO consists of a photonic integrated circuit containing a 9×3 photonic crossbar array with built-in modulators and photodiodes. The input weights are encoded by 9 electric absorption modulators same as the 27 crossbar weights. Only 5 input lanes and 2 output lanes (i.e., a 5x2 subsection) of the crossbar are used in the experiment. The output of the on-chip photodiodes is amplified by off-chip transimpedance amplifiers (FEMTO HSA-Y-1-60) and fed to an FPGA. Said FPGA (Xilinx HW-Z1-ZCU216-REVA03) controls the IPO by setting matrix and input weights as well as handling the data streams to an from the circuit via a custom Python API. The DACs driving the input EAMs operate at 4 GSa/s, the ADCs reading the photodiode currents operate at 2 GSa/s. Every input symbol gets expanded to four DAC samples and two consecutive ADC samples are combined into one symbol, resulting in a potential symbol rate of 1 GBd. To encode negative numbers balanced detection of the 2 crossbar outputs is performed in the FPGA reducing the matrix dimension to 1×5. Each modulator is calibrated to take crosstalk effects and non-linear weight scaling into account, calibration has been performed using the C34 input channel. The Output signal is scaled for every matrix according to an empirically determined rescaling factor based on an example MVM with 1280 random input vectors at an averaging factor of 2^8. The noise profiles used for simulating experimental noise levels were obtained from performing MVMs of 1280 random input vectors size 64 with a constant matrix of all weights equal to one. In supplementary information section S4, the power consumption in different sections of the demonstrator system is discussed.

FIGURES

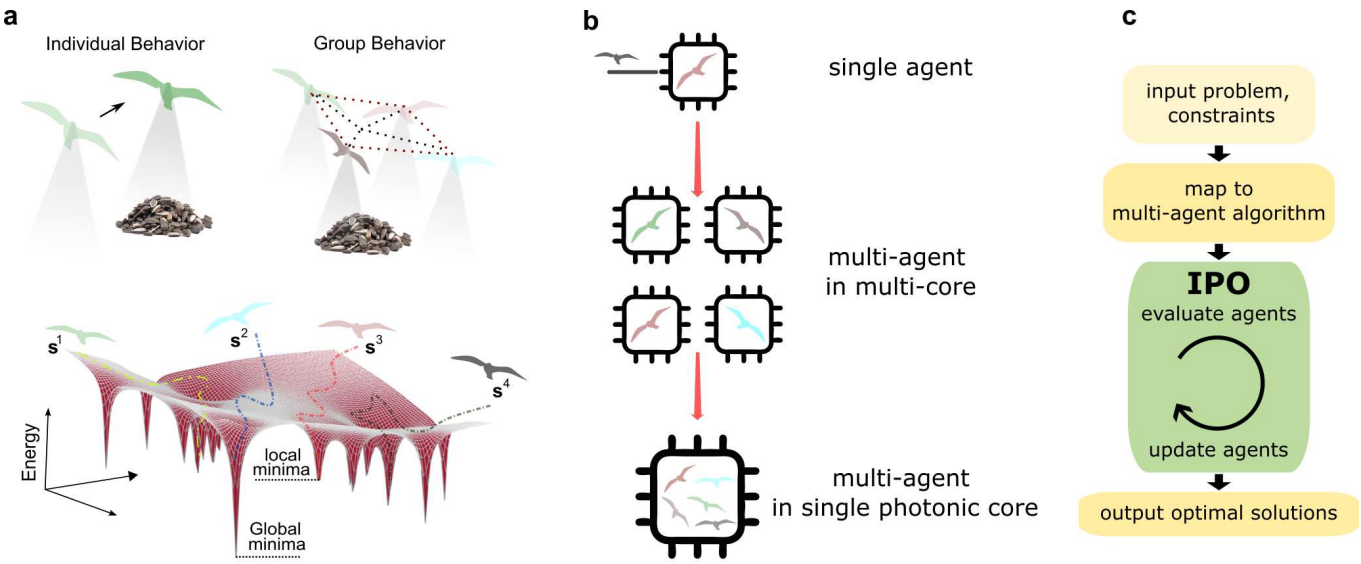


FIG. 1: **Concept of Photonic Optimizer**. **(a)** Illustration of nature inspired multi-agent optimization. A foraging avian must explore ($\vec{d}$) the landscape using their sensory range. The likelihood of finding the optimal food resource is improved through cooperation with a population of avians. When translated to the energy landscape of a typical NP-hard optimization problem, this strategy, leveraging the individual and social characteristics of multiple independent agents, can enable convergence to the global optimum in difficult optimization tasks. **(b)** Comparison of digital electronic architectures with the proposed photonic optimizer for solving multi-agent optimization problems. Digital electronics require many sequential processing steps, which can be executed on a single core or distributed across multiple cores. In contrast, the entire problem can be processed within a single photonic optimizer in a truly parallel, in-memory computing manner. **(c)** Flowchart describing the operational flows in the proposed integrated optimizer unit. When properly defined through chosen constraints, loss functions and update rules, the optimizer can iterate over a problem with multiple agents, evaluating all agents in a single cycle.

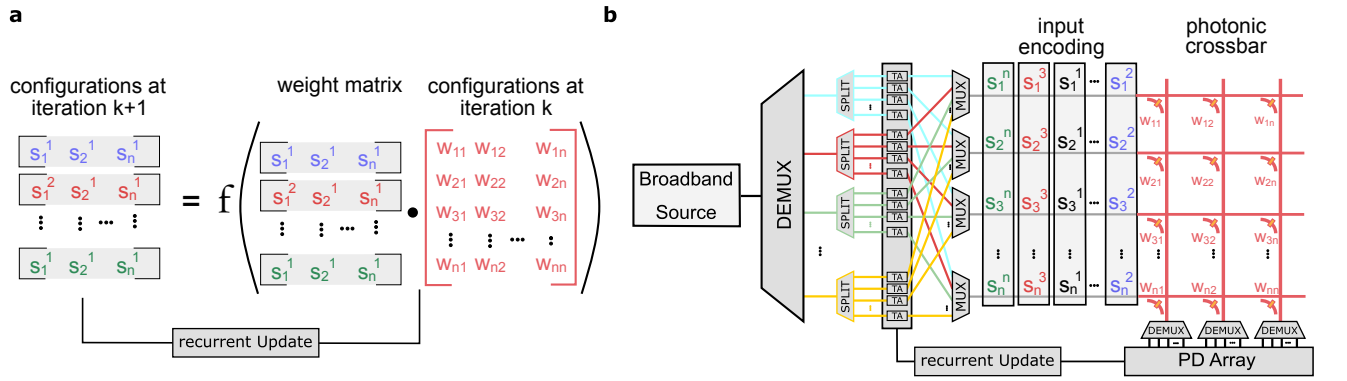


FIG. 2: **Conceptual Integrated Photonic Optimizer Unit**. **(a)** In an IPO unit, every agent is encoded as an input vector carried on a unique wavelength (optical carrier). The optimization problem is then represented and solved through matrix–vector operations, with the matrix elements encoded in the states of photonic memory devices. Multiple agents can be processed in parallel, leveraging both in-memory and state update computations to efficiently explore the solution space. **(b)** Conceptual schematic of the compute core. The core consists of multiple individually addressable wavelength channels (one per agent) and a high optical bandwidth linear processor. After demultiplexing a broadband light source (e.g., ASE source, laser array, or frequency comb), each wavelength channel is split into m distinct paths, each modulated by a tunable attenuator (TA) (e.g., EAM). This allows an input vector of length m to be encoded on each channel. The channels are then multiplexed into the m input waveguides of a photonic crossbar array, with each intersection also loaded with a tunable attenuator (e.g., phase-change material, EAM). The crossbar performs multiply-and-accumulate operations between matrix elements w_{ij} and all frequency channels in a single pass. The resulting vectors of are measured via demultiplexed photodetectors. The outputs of the photonic agent evaluation are then processed according to the chosen algorithm to update the inputs for the next cycle.

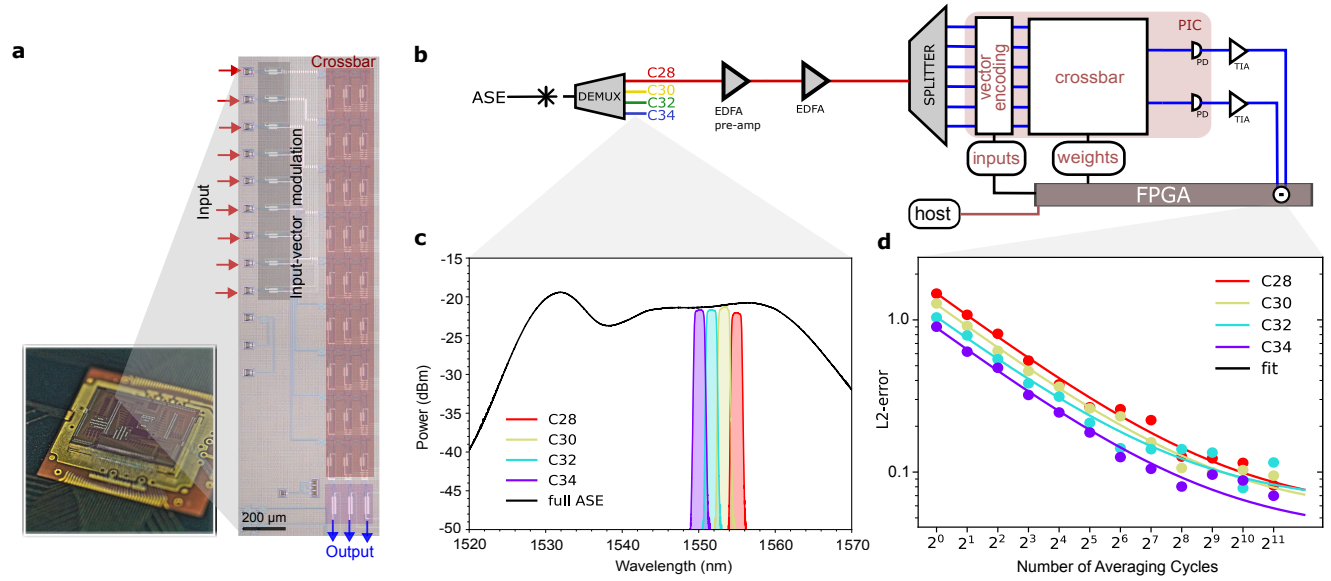


FIG. 3: **Prototype Integrated Photonic Optimizer Unit**. **a** Micrographs of a demonstrator silicon PIC. Various on-chip components are highlighted. **b** The IPO is evaluated for functionality and performance through a prototype hardware. Four distinct C-band wavelength channels (200GHz ITU grid channels C28, C30, C32 and C34) are spatially multiplexed to perform matrix-vector operations. A high-speed FPGA is interfaced with the photonic chip to handle auxiliary operations. Light from a broadband ASE source is split by a demultiplexer into the select channels, each subsequently amplified in two stages to roughly 100 mW before entering the silicon PIC. The PIC incorporates programmable input weights along with a 9×3 programmable crossbar array. Photocurrents from the on-chip photodetectors are amplified by off-chip TIAs and fed back into the same FPGA, which also controls the input signals and matrix weighting. **c** A plot depicting the spectrum of the ASE source along with the four distinct 200 GHz channels. **d** A plot showing L2-error of matrix-vector multiply operations for each input channel across averaging, with fitted noise profiles obtained through average error of 1280 random input vectors length 64 and a constant weight matrix.

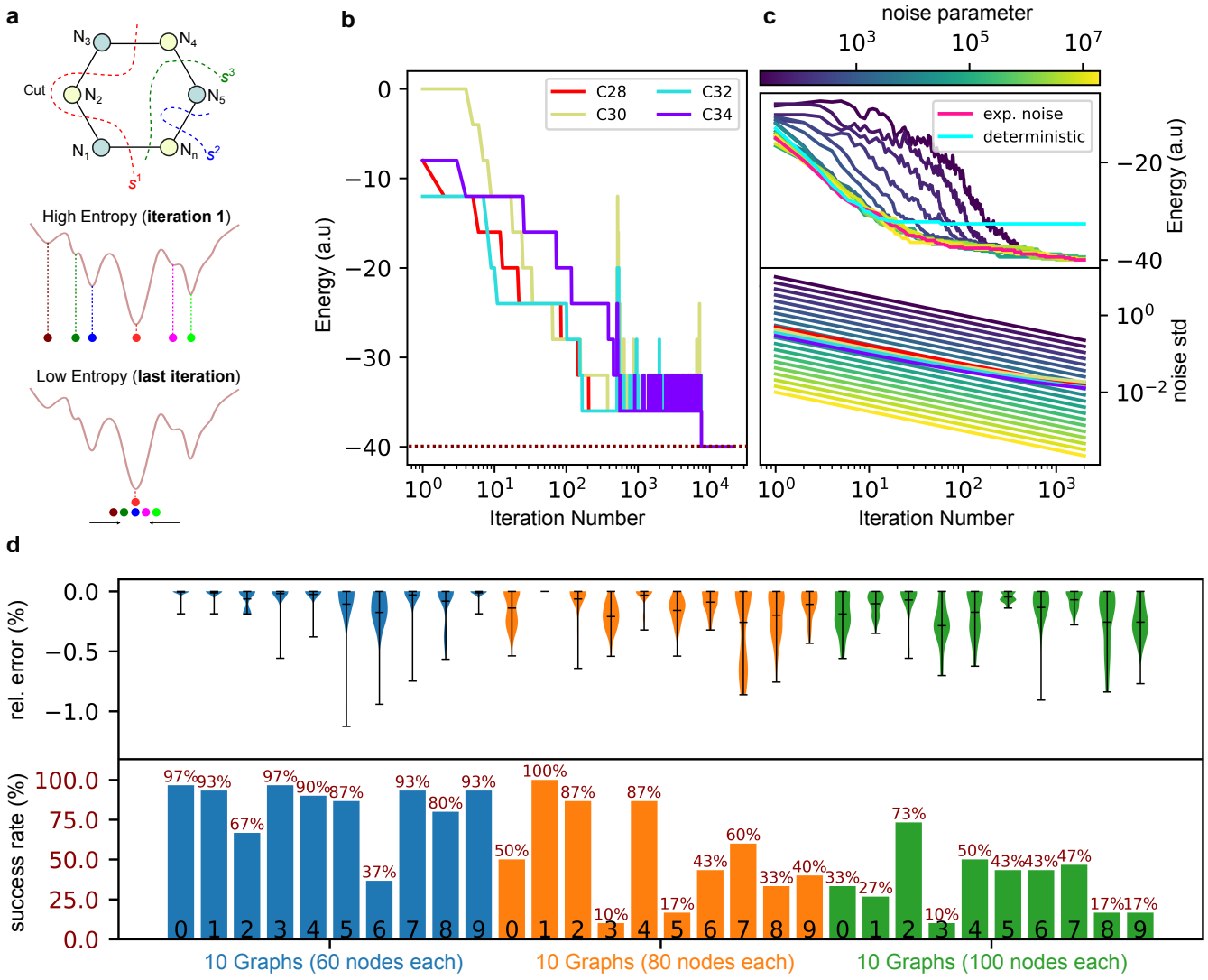


FIG. 4: **Combinatorial Optimization Problems**. (a) The top panel presents a schematic of the Max-Cut graph optimization problem solved by multiple agents. The lower panels show the energy landscape under the dynamic referencing algorithm, where multiple agents (colored dots) evolve in parallel and converge toward the global minimum, progressively reducing entropy. (b) Exemplary emulated optimization of a circular graph of 40 nodes using measured compute accuracy levels for the four ITU-channels from the benchmark measurement. The dashed red line cuts the graph into two sets of complementary nodes. The edges are encoded as matrix weights in the photonic crossbar. **(c)** Top Panel: Average global energy of twenty runs discovering the Max-cut of a circular graph of size 40 for different decaying noise profiles. The cyan line shows the average for purely deterministic calculations. The red line the performance using experimentally measured noise levels. While deterministic multi-agent approach can enable optimal convergence with increasing number of agents and iterations, we find optimal noise profiles to allow quicker convergence to the optimal solutions. **(d)** Performance Benchmarks on non circular graphs. Examples are taken from the Biq-Mac library (39). We use the experimentally measured noise profiles and four agents to solve each graph across 30 runs with different initial conditions. The success rate and relative errors are then computed from the fraction of runs that converge to the known solution. Even without deliberate hyperparameter tuning, every graph can be solved, with at least one successful trial in each case. In the top panel, the bars indicate the maximum, minimum, and mean of the relative error.

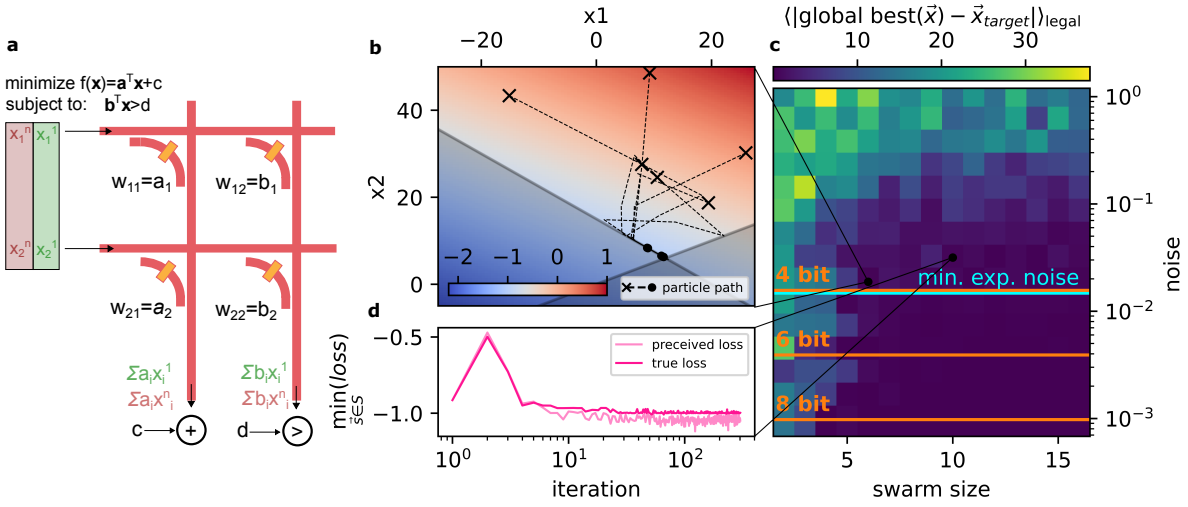


FIG. 5: **Linear Programming Problems**. **(a)** A schematic showing the configuration of a crossbar unit used for solving a constrained linear optimization problem through canonical particle swarm approach. Multiple queries encoded into the amplitude modulated optical signals operate in parallel. Evaluation of constraints are uniquely handled in the same pass by mapping into extra columns in the crossbar. **(b)** A snapshot of 6 agents' trajectories during balanced exploration and exploitation under moderate injected photonic noise in an emulated particle swarm optimization problem. The bottom panel shows evolution of the cost function as a function of iterations. **(c)** A plot showing the final average swarm position distance from the true minimum for canonical particle swarm problem for swarm sizes 2 to 16. The experimentally achievable noise levels and comparative approximate bit-accuracies are indicated.**(d)** The minimum swarm loss as a function of number of iterations for a swarm size 10 is shown. Here, the perceived loss is the loss recorded by the swarm through noisy evaluations, while true loss the exact loss at the resulting positions of each agent.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SIMain.pdf