NEVO: A Neuromorphic EVolutionary Optimiser with Spike-Driven Cortico-Basal-Thalamic Coordination

Jorge M. Cruz-Duarte and El-Ghazali Talbi
University of Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France.
E-Mails: {jorge.cruz-duarte, el-ghazali.talbi}@univ-lille.fr

***Abstract*—Neuromorphic computing and evolutionary computation share key properties, including distributed state, event-driven communication, and tolerance to finite precision. Nevertheless, optimisation control is rarely implemented directly in spiking dynamics. This paper introduces the Neuromorphic EVolutionary Optimiser (NEVO), in which evolutionary operator selection is realised as spike-driven action selection within a Cortico-Basal-Thalamic Loop (CBTL), whilst candidate generation and objective evaluation remain algorithmic under a standard black-box interface. NEVO is evaluated on the noiseless BBOB suite across 2D to 40D with 15 instances per function. Results show that spike-driven coordination sustains coherent operator sequences across problem landscapes, achieving successful target attainment in 2D and 3D, whilst exposing scalability limitations beyond 10D. These findings confirm that spike-driven coordination is feasible on neuromorphic substrates and clarify how neuromorphic action selection relates to existing adaptive operator selection strategies. The analysis also identifies concrete design requirements for future implementations.**

***Index Terms*—Neuromorphic Optimisation, Evolutionary Computation, Operator Selection, Spiking Neural Networks, Neural Engineering Framework, Nengo, Basal Ganglia.**

I. Introduction

ENERGY and latency constraints increasingly limit where optimisation can be deployed, particularly in embedded and autonomous settings where compute and memory are tightly budgeted. In von Neumann systems, the separation between processing and memory makes data movement a dominant cost. Iterative black-box optimisation amplifies this cost through repeated evaluations and population updates [1].

Evolutionary Computation (EC) remains effective for problems where gradients are unavailable or unreliable [2]. Its operators are local, its updates can be asynchronous, and its search tolerates finite precision. However, conventional implementations still rely on synchronous execution and repeated memory-access patterns that are not aligned with this structure.

Neuromorphic Computing (NC) provides event-driven architectures with collocated memory and computation. Platforms such as Intel Loihi, IBM TrueNorth, SpiNNaker, and BrainScaleS support sparse communication and distributed parallelism [3]–[7], whilst software frameworks such as Nengo implement the Neural Engineering Framework (NEF) for principled construction of spiking neural circuits [8].

NEVO is part of the NeurOptim research initiative: https://neuroptim.org.

Existing work supports neuromorphic optimisation, but it is commonly problem-bound or hybrid, leaving the optimisation loop outside the scope of spike-driven dynamics. A first strand exploits the device physics or hardwired setup. For instance, ferroelectric and organic memristor networks [9,10], RRAM spiking swarms [11], and Simulated Annealing on Loihi 2 [12] report promising efficiency in tackling combinatorial problems. Their designs are strongly coupled to a formulation or device. Thus, extending them to continuous, constrained, or multi-objective domains typically requires redesigning neural circuitry.

A second strand, often labelled neuroevolution, evolves spiking models but retains a conventional search process. The Evolutionary Optimisation for NC Systems framework [13] and related surveys [14]–[16] show that EC algorithms can configure SNN architectures and learning rules. Nevertheless, NC hardware is primarily used for inference or evaluation. Moreover, swarm-inspired approaches provide spiking interpretations of established Metaheuristics (MHs) [17,18]. These studies clarify how collective search can be expressed in spiking dynamics, but they rarely exploit event-driven execution, collocated memory and computation, and on-chip plasticity. Likewise, probabilistic optimisation has been integrated into NC platforms through hybrid pipelines. LavaBO [19,20] demonstrates scheduling and coordination within the Lava environment. The Bayesian core remains CPU-based. Recently, work has started to formalise the design space of neuromorphic heuristic search. Nheuristics [21] and NeurOptimiser [22] illustrate decentralised spike-triggered coordination. Current prototypes still rely on conventional code for candidate generation and objective evaluation.

This paper tests the hypothesis that evolutionary operator coordination can be realised as spike-driven action selection within a Cortico-Basal-Thalamic Loop (CBTL), while preserving a standard black-box interface for candidate generation and objective evaluation. The objective is to confirm that spike-driven operator coordination is feasible with these interfaces and to systematically identify the architectural and algorithmic constraints. Such phenomena emerge when MH control is delegated to spiking dynamics.

We propose the Neuromorphic EVolutionary Optimiser (NEVO), which uses a spike-driven CBTL controller to gate a collection of search operators from well-known MHs based

on a compact search state and adaptive operator weights. We then evaluate NEVO on the noiseless BBOB suite and report behavioural traces and systematic performance across multiple dimensions. We also analyse the dynamics of search operator selection and identify key limitations in operator weighting and utility design, which impede scalability beyond low dimensions.

II. Proposed Approach

This section presents NEVO, which implements operator selection via spike-driven CBTL whilst preserving a standard black-box interface. It maintains a solution archive in normalised domain $\mathfrak{V}$, computes state vector $\boldsymbol{s}(t)$ encoding diversity, improvement rate, and convergence, and maps weighted utilities $\boldsymbol{u}_o(t)$ via ε-greedy policy to action vector $\boldsymbol{a}(t)$ selecting the active operator.

Fig. 1 illustrates the coordination loop: the Sensory Cortex encodes $\boldsymbol{s}(t)$, the Striatum computes $\boldsymbol{u}_o(t)$, and the Basal Ganglia implement Winner-Take-All (WTA) competition to disinhibit a single thalamic channel. The Thalamus gates $\boldsymbol{a}(t)$, which activates the Motor Cortex to trigger the selected operator and update the population $\boldsymbol{X}(t)$. Feedback from fitness yields a reward signal $r(t)$ that updates the operator weights and closes the loop. The neuromorphic controller follows the NEF paradigm [23], which encodes continuous state variables in spiking ensembles and decodes their activity into control signals for online operator selection. The CBTL architecture is chosen over alternative action-selection mechanisms for three reasons: (i) it provides a biologically grounded WTA mechanism via mutual inhibition, avoiding the credit assignment problem inherent in temporal-difference methods; (ii) it operates in continuous time with distributed representations, enabling direct integration with event-driven NC hardware; and (iii) it supports online adaptation via local synaptic weight updates rather than requiring gradient backpropagation.

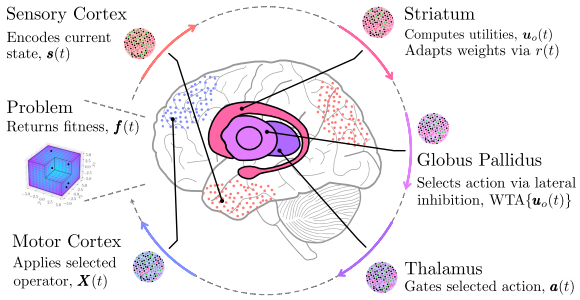


Fig. 1. NEVO architecture with a CBTL-based operator selection. The Sensory Cortex encodes the state $\boldsymbol{s}(t)$, the Striatum computes utilities $\boldsymbol{u}_o(t)$, and the Basal Ganglia implement WTA to gate an approximately one-hot action vector $\boldsymbol{a}(t)$ via the Thalamus. The Motor Cortex triggers the selected operator to generate $\boldsymbol{X}(t)$ and obtain fitness $f(t)$, and the resulting reward $r(t) \propto \Delta f(t)$ updates the memory and Striatum weights, closing the cycle.

A. Algorithmic Architecture

From an algorithmic perspective, NEVO follows the generalised MH model in [24] and adopts the low- and high-level view commonly used in automated algorithm design and configuration [25]–[27]. In this sense, NEVO can be seen as a basic state-based adaptive operator selection or hyper-heuristic controller [27,28], implemented as a spike-driven CBTL circuit. At the low level, heuristics consist of a collection of simple search operators that generate or modify candidate solutions in the problem domain. The high-level technique (or MH) selects which operator to apply at each iteration based on the current search state. In this work, we assume a bounded continuous minimisation problem and focus on coordination rather than problem formulation. We denote it by $(\mathfrak{X}, f, \min)$ with $\mathfrak{X} \subseteq \mathbb{R}^D$ and $f : \mathfrak{X} \to \mathbb{R}$, and seek $\boldsymbol{x}_* = \operatorname{argmin}_{\boldsymbol{x}\in\mathfrak{X}} f(\boldsymbol{x})$. At the high level, the MH coordinates a set of Simple Heuristics (SHs) $\mathfrak{H} \subseteq \mathbb{H}$ [24,25]. An SH is generative $h_g \in \mathfrak{H}$ or selective $h_s \in \mathfrak{H}$, and their composition defines a search operator, $h_o \triangleq h_s \circ h_g \in \mathfrak{H}_o \subset \mathfrak{H}$ [24]. At iteration k, the MH selects the h_o that maximises its utility under the current search state, $h_*^k = \operatorname{argmax}_{h_o\in\mathfrak{H}_o\subset\mathfrak{H}} \{U(h_o, \boldsymbol{s}^k)\}$, where $\boldsymbol{s}^k$ encodes diversity, improvement rate, and convergence, and U denotes the expected utility [1,29]. The selected h_o is applied as $\boldsymbol{X}^{k+1} \leftarrow h_*^k\{\boldsymbol{X}^k\}$, since $\boldsymbol{X}^k \in \mathfrak{X}^N$ stands for the k^{th} step of a population of N candidate solutions. The expression for h_*^k covers operator collection, where each decision cycle selects the operator with the highest state-dependent utility [30]. This selection is implemented via spike-driven action gating, whereas the reward updates the memory archive and operator weights.

1) Domain Transformation and Memory Management: Candidate solutions are stored in normalised domain $\mathfrak{V} = [-1, 1]^D$ and mapped to $\mathfrak{X} = [\boldsymbol{l}_b, \boldsymbol{u}_b]$ via affine transform $\mathcal{T} : \mathfrak{V} \to \mathfrak{X}$. NEVO maintains a fixed-size archive (memory) of capacity M, storing pairs $(\boldsymbol{v}_i, f_i)$ with empty slots marked by sentinel $f_{\text{worst}} = \max_i\{f_i\}$. Each candidate $(\boldsymbol{v}_{\text{new}}, f_{\text{new}})$ replaces the worst entry when $f_{\text{new}} < f_{\text{worst}}$, and age counters track entry persistence.

2) State Feature Extraction: We consider a search state given by $\boldsymbol{s}^k = (\phi_d^k, \phi_i^k, \phi_c^k)^\intercal \in [0, 1]^3$, computed from the current archive $\boldsymbol{V}^k$ and fitness vector $\boldsymbol{f}^k$. These state features correspond respectively to diversity, improvement rate, and convergence, defined as

$$\begin{aligned} \phi_d^k &= \sqrt{3}D^{-1} \textstyle\sum_{j=1}^{D} \operatorname{Std}\{\boldsymbol{v}_{:,j}^k\}, \\ \phi_i^k &= W^{-1} \textstyle\sum_{q=k-W+1}^{k} \mathbb{I}(\, f_\star^q < f_\star^{q-1}), \\ \phi_c^k &= \left(1 + \lambda_c \Delta f^k / (\,|f_\star^k| + \eta_c\, \Delta f^k + \epsilon)\right). \end{aligned} \tag{1}$$

From these, $\boldsymbol{v}_{:,j}^k = (v_{1,j}^k, \ldots, v_{N,j}^k)^\intercal$ is the j^{th} column of $\boldsymbol{V}^k$, $\mathbb{I}(\cdot)$ is the indicator function, and the normalisation factor $1/\sqrt{3}$ comes from $\operatorname{Std}\{\mathcal{U}(-1, 1)\}$. The best-so-far fitness value is $f_\star^k = \min_i\{f_i^k\}$, achieved at $\boldsymbol{v}_\star^k$. Moreover, $\Delta f^k = \max(\boldsymbol{f}^k) - \min(\boldsymbol{f}^k)$, with λ_c $(= 10)$, η_c $(= 10^{-2})$, and ϵ $(= 10^{-12})$ controlling sensitivity and numerical stability. Plus, valid archive entries satisfy $f_i < f_{\text{worst}}$, as defined by the sentinel value. When fewer than two valid solutions exist, we set $\boldsymbol{s}^k = (0.5, 0.5, 0.0)^\intercal$, to reflect an uninformative initial condition and no evidence of convergence.

3) Operator Collection: NEVO employs thirteen search operators from well-known MHs. Exploration operators are:

Random Search (RS), Lévy Flight (LF), Genetic Crossover (GX), Differential Evolution (DE), Central Force Optimisation (CF), Firefly Algorithm (FA), and Gravitational Search (GS) [1,26,31]–[34]. Correspondingly, exploitation operators are: Local Random Walk (LW), Genetic Mutation (GM), Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimisation (PS), and Spiral Optimisation (SO) [1,26,35]–[37]. These labels are used only as a coarse description of the intended role of each implementation and do not imply a strict dichotomy, since the actual exploratory or exploitative behaviour of a given operator depends on the current population and problem landscape. Each operator generates N candidates around fitness-weighted centroid $\boldsymbol{v}_{\odot}^{k}$, some considering the current memory $\boldsymbol{V}^{k}$. For additional details, refer to [38].

4) Utility Modelling and Weight Adaptation: Each operator has utility $u_o : \mathbb{R}^3 \to \mathbb{R}$ mapping state $\boldsymbol{s}$ to scalar score, and adaptive weight w_{h_o} yielding weighted utility $\tilde{u}_{h_o}(\boldsymbol{s}) = w_{h_o} u_{h_o}(\boldsymbol{s})$. We deliberately use a linear map with fixed $(\boldsymbol{A}, \boldsymbol{b})$ as the simplest baseline compatible with the CBTL and NEF implementation. The weights are initialised uniformly at $w_{h_o}(0) = 1.0$. Thus, utilities are computed as $\boldsymbol{u}_o^k = \boldsymbol{W}\,(\boldsymbol{A}\,\boldsymbol{s}^{k+1} + \boldsymbol{b})$, where $\boldsymbol{W} = \mathrm{diag}(w_{h_{\mathrm{RS}}}, \ldots, w_{h_{\mathrm{SO}}})$. After each evaluation, reward $r^k = \max\{0,\, f_\star^{k-1} - f_\star^k\}/(|f_\star^{k-1}| + \epsilon)$ updates weights via $w_{h_o}^k \leftarrow \min\{w_{\max}, \max\{w_{\min}, w_{h_o}^{k-1} + \eta r^k - \delta(1 - \mathbb{I}(r^k > 0))\}\}$ with $\eta = 0.4$, $\delta = 0.01$, and bounds $[0.1, 5.0]$. Action selection applies ε-greedy policy ($\varepsilon = 0.1$) over basal ganglia output [39].

B. Cortico-Basal-Thalamic Loop Architecture

Operator selection is implemented in Nengo [8] as a CBTL circuit: a state ensemble encodes $\boldsymbol{s}(t)$, utility ensembles (one per operator) compute $u_o(\boldsymbol{s}(t))$, basal ganglia WTA network performs competition, and thalamus gates action vector $\boldsymbol{a}(t) \in \mathbb{R}^{13}$ for operator selection [40,41]. The closed loop $\boldsymbol{s}(t) \to \boldsymbol{u}_o(t) \to \boldsymbol{a}(t) \to \boldsymbol{X}(t) \to r(t)$ updates every Δt_{eval}.

C. Complexity Analysis

Per decision cycle, the dominant cost is objective evaluation at $\mathcal{O}(NC_f)$. Candidate generation ranges from $\mathcal{O}(ND)$ for simple perturbation-based operators to $\mathcal{O}(NM \log M)$ for parent selection with sorting, $\mathcal{O}(N^2 D)$ for distance-based ones, and $\mathcal{O}(ND^2)$ for coordinate plane rotations. State feature extraction costs $\mathcal{O}(MD)$ for diversity and convergence computations, archive updates cost $\mathcal{O}(NM)$ via linear scanning of the memory, and utility computation costs $\mathcal{O}(K)$ for K operators. The neuromorphic controller simulates E neurons over $\Delta t_{\mathrm{eval}}/\Delta t$ steps per decision cycle, contributing $\mathcal{O}((E + C)\,\Delta t_{\mathrm{eval}}/\Delta t)$, where C counts the active synapses.

Combining these terms renders a total per-cycle cost of $\mathcal{O}(NC_f + C_{\mathrm{o}}(N, D, M) + MD + NM + K + (E + C)\,\Delta t_{\mathrm{eval}}/\Delta t)$, where $C_{\mathrm{o}}(N, D, M)$ ranges from $\mathcal{O}(ND)$ to $\mathcal{O}(N^2 D)$ or $\mathcal{O}(NMD)$ depending on the operator. When C_f dominates, the asymptotic cost reduces to $\mathcal{O}(NC_f)$. However, the neural simulation term $\mathcal{O}((E + C)\,\Delta t_{\mathrm{eval}}/\Delta t)$ represents a fixed overhead that is independent of problem dimensionality and can become significant in software-based implementations. On dedicated NC hardware with event-driven asynchronous execution, this overhead would be substantially reduced, shifting the bottleneck to objective evaluation or operator-specific generation costs. It then promotes improved scalability and supports embedded real-time optimisation.

III. Experimental Evaluation

A. Setup and Protocol

Simulations used Nengo v4.1.0 [8] with LIF neurons on BBOB noiseless suite from COCO v2.8.2 [42], and IOHexperimenter v0.3.18 [43]. These were executed in Python v3.10 on an AMD-Penguin Computing with AMD EPYC 7642 (48 cores, 512 GB RAM), part of the Grid'5000[1]. We evaluated 24 problems across six dimensions (2, 3, 5, 10, 20, 40) with 15 instances each, grouped as separable (`separ`, f1-f5), unimodal low- and high-conditioning (`lcond`, f6-f9; `hcond`, f10-f14), and multimodal adequate and weak structure (`multi`, f15-f19; `mult2`, f20-f24). For each problem-instance-dimension configuration, we performed a single run, with 15 trials per function in accordance with [42]. For NEVO setup, we used $N = 50$, $N_{\mathrm{neurons}} = 50$, $M = 25$, and $\Delta t = 1$ ms. Further details are freely available in the repository at [38].

B. Preliminary Validation

In the first experiment, we used the first instance of the first function in each BBOB category, using 10D and 5 s of simulation time. Fig. 2 shows representative dynamics on f01 (Sphere) and f10 (Ellipsoidal rotated). On f01, the CBTL controller yields smooth convergence and shifts from exploratory operators (LF, GX, and RS) to exploitation-focused ones (TS, GM, and SO) as ϕ_d decreases and ϕ_c increases. Diversity decreases, ϕ_i remains active, and ϕ_c rises as the population concentrates near the optimum. On f10, improvements are sporadic, and convergence stabilises late. The operator usage pattern remains almost unchanged, yet the state features indicate sustained exploration and repeated attempts to accommodate the landscape.

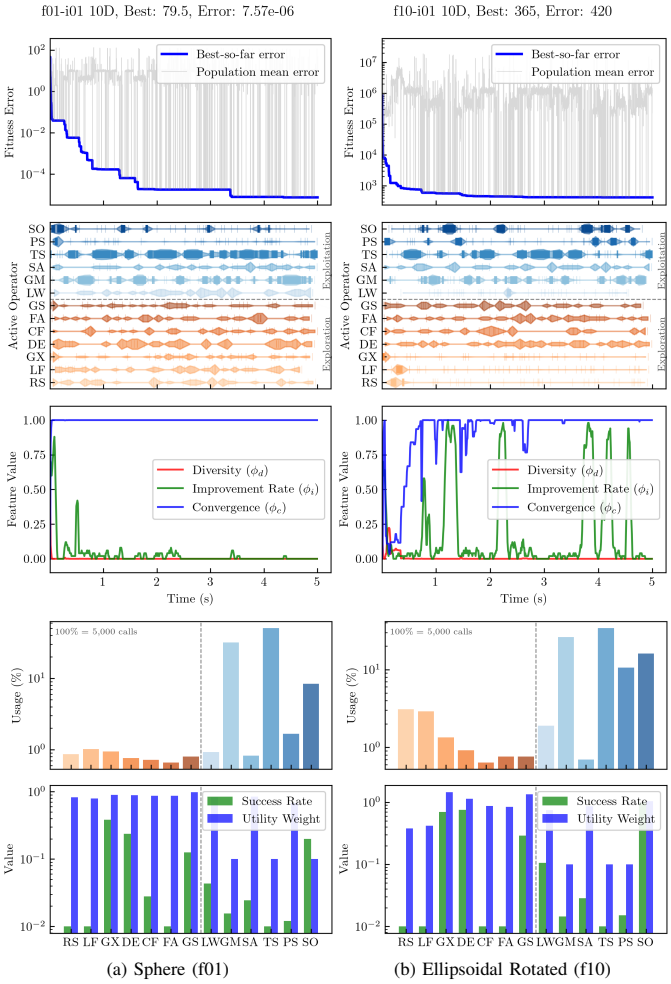
These traces indicate that NEVO uses spike-driven operator selection to maintain coherent coordination within the NEF architecture. The contrast between f01 and f10 highlights sensitivity to landscape structure and the limits of a fixed utility mapping, which motivates the analysis that follows.

C. Systematic Benchmark Results

In the confirmatory experiment, we systematically evaluated NEVO across all benchmark functions, instances, and dimensions of the BBOB noiseless suite. Each configuration was run using 20 s as simulation time. In such a way, we ensured sufficient evaluations for robust statistical analysis and direct comparison with results from the COCO repository [42].

Fig. 3 reports bootstrapped Empirical Cumulative Distribution Functions (ECDFs) for 51 target precision levels across the considered dimensions. ECDFs quantify the fraction of targets reached as a function of the evaluation budget, aggregated

[1]Further information about the Grid'5000 project: https://www.grid5000.fr.



(a) Sphere (f01) (b) Ellipsoidal Rotated (f10)

Fig. 2. Representative single-run optimisation dynamics of the CBTL architecture on (a) Sphere (f01) and (b) Ellipsoidal Rotated (f10) in 10D. First row: Convergence of best-so-far fitness error with population mean error envelope. Second row: Operator selection timeline showing active operator switches. Third row: Evolution of search-state feature encodings for diversity, improvement rate, and convergence. Fourth row: Operator usage counts aggregated over time. Fifth row: Operator success rates and utility weights evolution. Search operators are organised as described in Section II-A3.

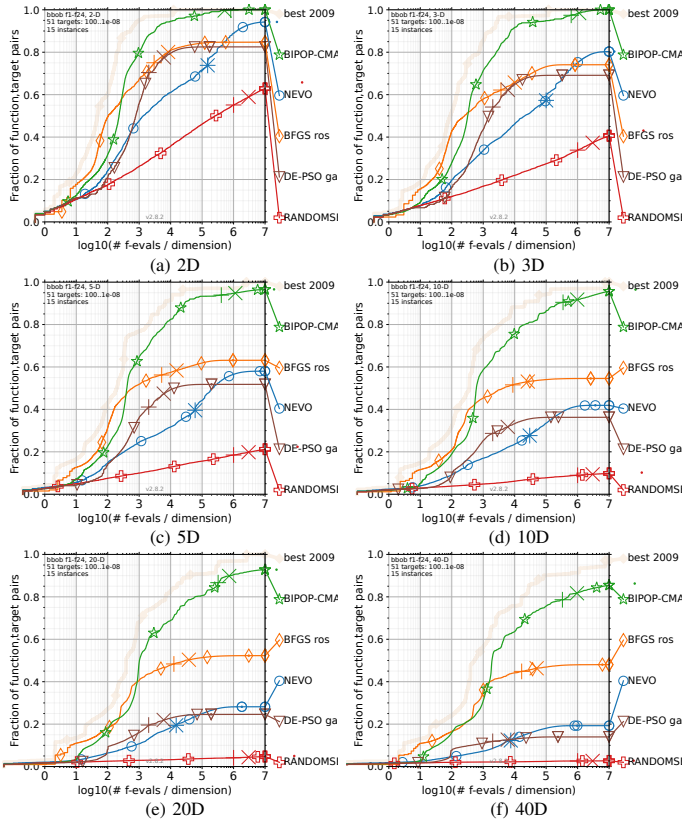


Fig. 3. Bootstrapped Empirical Cumulative Distribution Functions (ECDFs) of the number of f-evaluations divided by dimension for 51 targets in $10^{[-8..2]}$ across all function subgroups, with 15 instances per dimension, shown for dimensions 2, 3, 5, 10, 20, and 40. The best 2009 (upper bound ECDFs) is shown as a light, thick reference line. Legend: ◯: NEVO (this work), ☆: BIPOP-CMA-ES hansen [44], ♢: BFGS ros [45], ▽: DE-PSO garcia-nieto [46], ✚ : RANDOMSEARCH auger [31]. Comparison datasets were obtained from COCO [42]. Data produced with COCO v2.8.2

across problems and instances [42]. In 2D and 3D, NEVO performs well on separable and moderately conditioned functions, with curves close to BIPOP-CMA-ES on `separ` and `lcond` under moderate budgets. In 5D, performance degrades on multimodal and weakly structured functions. From 10D onwards, the curves flatten at low target attainment fractions, indicating that most runs do not reach the targets within the available budget. In 20D and 40D, performance is comparable to DE-PSO and remains above RANDOMSEARCH.

Fig. 4 depicts mean operator weight vs mean success rate. The moderate correlation ($r = 0.559$) indicates a partial alignment between the weight updates and the observed improvements. Several operators attain high weight and high success, and exploration operators, marked with squares, tend to receive higher weights than exploitation operators. This is consistent with the reward (cf. Section II-A4), which emphasises immediate and substantial improvements. Other operators achieve high weights despite low success, which indicates that frequent selection can be rewarded even when improvement events are rare. SO and PS illustrate this imbalance, as they achieve high or moderate success under low or moderate weights. The current rule, therefore, overvalues sporadic large improvements and undervalues consistent incremental progress, while remaining insensitive to diversity and delayed benefits.

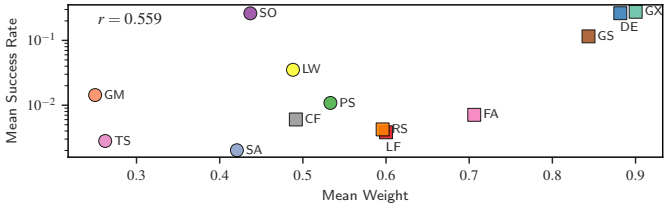


Fig. 4. Relationship between mean operator weight and mean success rate. Each point represents an operator, with colour indicating operator identity. r corresponds to the Pearson correlation coefficient.

Subsequently, Fig. 5 details operator weights across dimensions and BBOB categories. In 2D and 3D, exploration operators (except CF) consistently receive weights greater than 0.52 across categories, whereas certain exploitation operators

(SO, TS, GM) remain suppressed below 0.4. PS and SA occupy intermediate positions (0.4-0.6). As dimensionality increases (5D-10D), the same exploration-dominant pattern persists, with GX, DE, and GS surpassing the 0.7 mark, while the exploitation ones remain near their initial values. In 20D and 40D, exploration operator weights inflate substantially, with GX, DE, and GS exceeding 1.0 and reaching 1.3-1.5 on certain categories, while exploitation operators remain below 0.5. This weight inflation does not correspond to improved performance; the ECDFs in Fig. 3 show substantial degradation beyond 10D. The pattern reveals that the reward mechanism systematically favours exploration operators that occasionally produce large improvements, whereas exploitation operators that deliver incremental refinement are consistently undervalued. These patterns impede effective coordination between the exploration and exploitation phases across all dimensions.

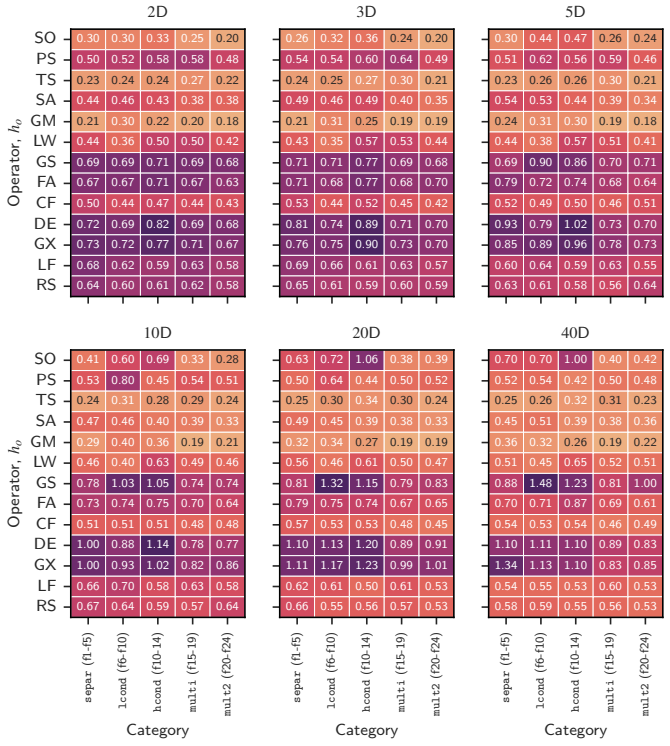
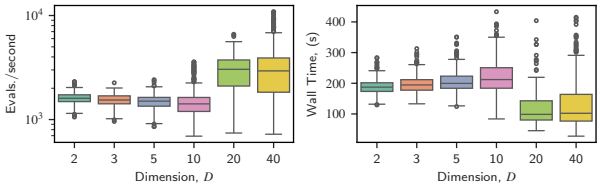


Fig. 5. Operator weight heatmaps faceted by problem dimension, showing how operator preferences vary across BBOB problem categories. Each heatmap corresponds to a different problem dimensionality (2D-40D).

Finally, Fig. 6 summarises computational efficiency across the experimental setup. Evaluation throughput (Fig. 6a) decreases from approximately 10^4 evaluations per second in 2D to 10^3 in 40D. It is consistent with the $\mathcal{O}(ND)$ scaling of candidate generation and state feature extraction. Besides, the wall-clock time per experiment (Fig. 6b) duplicates its value (from about 200 s to 400 s) when the dimensionality increases from 2 to 40. It reflects both the fixed 20-second simulation time and the overhead of the NC controller simulation. Moreover, evaluating the objective function is the dominant computational cost throughout, while neural simulation overhead remains relatively constant and dimension-dependent operations contribute sublinearly to the total execution time.



(a) Evaluation Throughput (b) Wall-clock Time

Fig. 6. Performance summary by problem dimensionality: (a) Number of evaluations per second, and (b) Wall-clock time required per experiment.

D. Discussion

Spike-driven operator selection via CBTL successfully coordinates search operators under a standard black-box interface. Strong 2D-3D target attainment and acceptable 5D performance validate the approach, with structured operator sequences that track state evolution. Performance degrades beyond 10D, with partial weight-success alignment ($r = 0.559$) and systematic exploration-operator bias visible across dimensions. We notice that rewarding immediate improvements undervalues incremental progress and creates an imbalance between the exploration and exploitation phases.

Results reflect fixed resources ($N = 50$, $M = 25$, 20 s simulation) and hybrid implementation (algorithmic candidate generation and evaluation). The experiments establish the feasibility of spike-driven coordination whilst identifying concrete limitations: scalability, reward design that balances exploration and exploitation, and neuromorphic representations for solutions and memory.

IV. Conclusions and Future Work

We introduced NEVO, an NC-based evolutionary optimiser in which operator selection is implemented via spike-driven coordination within a CBTL, realised through the NEF paradigm. NEVO realises evolutionary operator coordination as a spike-driven control loop and validates this mechanism on a standard black-box benchmark suite, verifying its feasibility for neuromorphic evolutionary optimisation.

Our experiments on the noiseless BBOB suite revealed that spike-driven action selection can sustain coherent operator sequences and achieve strong target attainment in 2D and 3D, with acceptable behaviour in 5D. Performance deteriorates from 10D forth under the fixed-resource setting used in this study. The weight statistics indicate a systematic bias toward exploration operators, driven by a reward function that primarily responds to immediate improvements in best fitness.

Future work should address scalability by revising the reward and weight updates to account for incremental progress, population-level improvements, and diversity maintenance. The utility mapping should be learned online from the state and outcomes rather than fixed, and the state encoding should be extended to capture landscape anisotropy and rotation sensitivity without increasing decoding burden. A second avenue is to reduce hybridity by developing NC representations for

solution memory, candidate generation, and selection, and by mapping these mechanisms to deployable substrates. Finally, evaluation protocols should jointly report solution quality, latency, and energy, and should include hardware-backed measurements to quantify the benefits of spike-driven optimisation.

Acknowledgements

Grammarly and Claude Sonnet 4.6 were used solely for language polishing under human supervision. The authors remain fully responsible for the scientific content of this manuscript.

References

[1] A. Eiben and J. Smith, "Introduction to evolutionary computing," *Assembly Automation*, vol. 24, no. 3, pp. 324–324, 2004.

[2] J. Liu, R. Sarker, S. Elsayed *et al.*, "Large-scale evolutionary optimization: A review and comparative study," *Swarm and Evolutionary Computation*, vol. 85, p. 101466, 2024.

[3] M. Davies, A. Wild, G. Orchard *et al.*, "Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, 2021.

[4] F. Akopyan, J. Sawada, A. Cassidy *et al.*, "Truenorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[5] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[6] C. Pehle, S. Billaudelle, B. Cramer *et al.*, "The brainscales-2 accelerated neuromorphic system with hybrid plasticity," *Frontiers in Neuroscience*, vol. 16, p. 795876, 2022.

[7] P. K. Enuganti, B. Sen Bhattacharya, T. Serrano Gotarredona, and O. Rhodes, "Neuromorphic computing and applications: a topical review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 15, no. 2, p. e70014, 2025.

[8] C. Eliasmith and T. Stewart, "Nengo and the neural engineering framework: Connecting cognitive theory to neuroscience," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.

[9] Z. Lin and Z. Fan, "A ferroelectric memristor-based transient chaotic neural network for solving combinatorial optimization problems," *Symmetry*, vol. 15, no. 1, p. 59, 2022.

[10] H. Kim, M. Kim, A. Lee *et al.*, "Organic memristor-based flexible neural networks with bio-realistic synaptic plasticity for complex combinatorial optimization," *Advanced Science*, vol. 10, no. 19, p. 2300659, 2023.

[11] A. S. Lele, M. Chang, S. D. Spetalnick *et al.*, "Neuromorphic swarm on RRAM compute-in-memory processor for solving QUBO problem," *2023 60th ACM/IEEE Design Automation Conference (DAC)*, p. 6, 2023.

[12] A. Pierro, P. Stratmann, G. A. F. Guerra *et al.*, "Solving QUBO on the Loihi 2 Neuromorphic Processor," *arXiv*, 2024.

[13] C. D. Schuman, J. P. Mitchell, R. M. Patton *et al.*, "Evolutionary optimization for neuromorphic systems," in *Proceedings of the 2020 Annual Neuro-Inspired Computational Elements (NICE) Workshop*, 2020.

[14] S. Shen, R. Zhang, C. Wang *et al.*, "Evolutionary spiking neural networks: A survey," *Journal of Membrane Computing*, vol. 6, no. 4, pp. 335–346, 2024.

[15] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[16] D. Kudithipudi, C. Schuman, C. M. Vineyard *et al.*, "Neuromorphic computing at scale," *Nature*, vol. 637, no. 8047, p. 801–812, 2025.

[17] Y. Fang, Z. Wang, J. Gomez, S. Datta, A. I. Khan, and A. Raychowdhury, "A swarm optimization solver based on ferroelectric spiking neural networks," *Frontiers in Neuroscience*, vol. 13, p. 855, 2019.

[18] T. Sasaki and H. Nakano, "Swarm intelligence algorithm based on spiking neural-oscillator networks, coupling interactions and search performances," *Nonlinear Theory and Its Applications, IEICE*, vol. 14, no. 2, p. 267–291, 2023.

[19] S. Snyder, S. R. Risbud, and M. Parsa, "Neuromorphic bayesian optimization in Lava," in *Proc. International Conference on Neuromorphic Systems (ICONS)*, 2023.

[20] S. Snyder, S. Risbud, and M. Parsa, "Asynchronous neuromorphic optimization in Lava," in *Proc. Great Lakes Symposium on VLSI*, 2024, p. 776–778.

[21] E. Talbi, "Neuromorphic-based metaheuristics: A new generation of low power, low latency and small footprint optimization algorithms," 2025. [Online]. Available: https://arxiv.org/abs/2505.16362

[22] J. M. Cruz-Duarte and E.-G. Talbi, "Neuroptimisation: The spiking way to evolve," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2507.08320

[23] C. Eliasmith and C. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2003.

[24] J. M. Cruz-Duarte, J. C. Ortiz-Bayliss, I. Amaya *et al.*, "Towards a generalised metaheuristic model for continuous optimisation problems," *Mathematics*, vol. 8, no. 11, p. 2046, 2020.

[25] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, vol. 285, pp. 405–428, 2020.

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

[27] A. Hassan and N. Pillay, "Dynamic heuristic set selection for cross-domain selection hyper-heuristics," in *International Conference on the Theory and Practice of Natural Computing*. Springer, 2021, pp. 33–44.

[28] J. Pei, Y. Mei, J. Liu, M. Zhang, and X. Yao, "Adaptive operator selection for meta-heuristics: A survey," *IEEE Transactions on Artificial Intelligence*, 2025.

[29] B. Doerr and C. Doerr, "Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices," *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, p. 271, 2019.

[30] L. Velasco, H. Guerrero, and A. Hospitaler, "A literature review and critical analysis of metaheuristics recently developed," *Archives of Computational Methods in Engineering*, vol. 31, pp. 125–146, 2024.

[31] A. Auger and R. Ros, "Benchmarking the pure random search on the bbob-2009 testbed," in *Proc. 11th Annual GECCO Conference: Late Breaking Papers*, 2009, pp. 2479–2484.

[32] J. Li, Q. An, H. Lei *et al.*, "Survey of lévy flight-based metaheuristics for optimization," *Mathematics*, vol. 10, no. 15, 2022.

[33] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[34] R. Formato, "Central force optimization: A new deterministic gradient-like optimization metaheuristic," *Opsearch*, vol. 46, pp. 25–51, 2009.

[35] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," in *Handbook of metaheuristics*. Springer, 2018, pp. 1–35.

[36] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. on Neural Networks (ICNN)*, vol. 4. ieee, 1995, pp. 1942–1948.

[37] K. Tamura and K. Yasuda, "Spiral dynamics inspired optimization," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 15, no. 8, pp. 1116–1122, 2011.

[38] J. M. Cruz-Duarte and E.-g. Talbi, "NEVO: A Neuromorphic EVolutionary Optimiser with Spike-Driven Cortico-Basal-Thalamic Coordination - Codes and Results," Mar. 2026. [Online]. Available: https://doi.org/10.5281/zenodo.19188034

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[40] D. Joseph and R. S. Bapi, "What do the basal ganglia do? A modeling perspective," *Biological cybernetics*, vol. 103, no. 3, pp. 237–253, 2010.

[41] S. Sharma, S. Aubin, and C. Eliasmith, "Large-scale cognitive model design using the nengo neural simulator," *Biologically Inspired Cognitive Architectures*, vol. 17, pp. 86–100, 2016.

[42] N. Hansen, A. Auger, R. Ros *et al.*, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, pp. 114–144, 2021.

[43] J. de Nobel, F. Ye, D. Vermetten *et al.*, "IOHexperimenter: Benchmarking Platform for Iterative Optimization Heuristics," *arXiv*, Nov. 2021. [Online]. Available: https://arxiv.org/abs/2111.04077

[44] N. Hansen, "Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed," in *Proc. 11th Annual GECCO Conference: Late Breaking Papers*, 2009, p. 2389–2396.

[45] R. Ros, "Benchmarking the BFGS algorithm on the BBOB-2009 function testbed," in *Proc. 11th Annual GECCO Conference: Late Breaking Papers*, 2009, p. 2409–2414.

[46] J. García-Nieto, E. Alba, and J. Apolloni, "Noiseless functions black-box optimization: evaluation of a hybrid particle swarm with differential operators," in *Proc. 11th Annual GECCO Conference: Late Breaking Papers*, 2009, p. 2231–2238.