



# NeurOptimiser: A General Framework for Neuromorphic Optimisation

Jorge M. Cruz-Duarte • El-Ghazali Talbi



# Introduction I

- The surge in AI workloads demands firm and scalable energy<sup>1</sup>, or **low-power, decentralised, and efficient** alternatives<sup>2</sup>.
- Spiking Neural Networks (SNNs) have gained attention as a biologically plausible paradigm for Neuromorphic Computing (NC).
  - Why? Because their **asynchronous, event-based, and local dynamics**  
→ Suitable for **decentralised and energy-efficient** systems<sup>3</sup>.
  - These properties are primarily relevant for embedded and edge applications.  
**Modern technology needs!**
- It motivates the **integration** of **optimisation** processes into **neuromorphic substrates**<sup>4</sup>.

---

<sup>1</sup>Engineering News-Record, *Power hungry: ai-fueled data center boom sets energy delivery's new course*, <https://www.enr.com/articles/61083-power-hungry-ai-fueled-data-center-boom-sets-energy-deliverys-new-course>, Accessed: 2025-07-30, 2025.

<sup>2</sup>D. Kudithipudi et al., "Neuromorphic computing at scale", *Nature* **637**, 801–812 (2025).

<sup>3</sup>M. Davies et al., "Loihi: a neuromorphic manycore processor with on-chip learning", *IEEE Micro* **38**, 82–99 (2018).

<sup>4</sup>E. Talbi, *Neuromorphic-based metaheuristics: a new generation of low power, low latency and small footprint optimization algorithms*, 2025.



# Introduction II

- Some efforts in the literature:
  - Bayesian optimisers compatible with Intel's Lava<sup>1</sup> (targeting Loihi devices).  
→ But they delegate decision-making to CPU-based components.
  - Swarm intelligence approaches simulating collective behaviour via SNNs<sup>2</sup>.  
→ But they lack modular heuristic rules or local adaptation.
  - GPU-based models prioritise simulation throughput<sup>3</sup>.  
→ But do not encode the optimisation process within spike-driven transitions.
- While these models rely on spiking hardware, their optimisation logic remains rooted in classical computation<sup>4</sup>. They are neuromorphic only at the infrastructural level.

<sup>1</sup>S. Snyder et al., "Asynchronous neuromorphic optimization in Lava", in Proc. great lakes symposium on vlsi 2024, GLSVLSI '24 (2024), pp. 776–778.

<sup>2</sup>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 **14**, 267–291 (2023).

<sup>3</sup>B. Golosio et al., "Fast simulations of highly-connected spiking cortical models using GPUs", Frontiers in Computational Neuroscience **15**, 627620 (2021).

<sup>4</sup>Sanaullah et al., "Exploring spiking neural networks: a comprehensive analysis of mathematical models and applications", Frontiers in Computational Neuroscience **17**, 1215824 (2023).



# Introduction III

That is why...

- We present the Neuromorphic Optimiser (NeurOptimiser).  
This is a general metaheuristic framework in which optimisation arises from spike-based local computation.
- This framework is the first to embed coordination and evolutionary search within spiking activity.
- It contributes to the Neuromorphic-based Metaheuristic (**Nheuristic**) paradigm<sup>1</sup>.

<sup>1</sup>E. Talbi, *Neuromorphic-based metaheuristics: a new generation of low power, low latency and small footprint optimization algorithms*, 2025.



## Introduction IV\* - Key Concepts

### Minimisation Problem

We define the *minimisation problem*  $(\mathfrak{X}, f, \min)$  as  $\mathbf{x}_* \in \mathfrak{X}$ , such as  $\mathbf{x}_* = \operatorname{argmin}_{\mathbf{x} \in \mathfrak{X}} \{f(\mathbf{x})\}$ , where  $\mathbf{x}_* \in \mathfrak{X}$  is the optimal solution and  $\mathfrak{X} \subseteq \mathbb{R}^d$  is the feasible domain.

### Population and Neighbourhood

A *population*  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$  consists of a finite set of  $n$  candidate solutions.

A *neighbourhood* is defined as  $\mathbf{N} \subseteq \mathbf{A}$ , since  $\mathbf{A}$  is an arbitrary set; e.g.,  $\mathbf{A} = \mathbf{X}, \mathbf{A} = \mathbf{N}_i^t, \mathbf{A} = \mathbf{P}^t$ . (Considering that  $\mathbf{N}_i^t = \{\mathbf{x}_i^\tau\}_{\tau=1}^t$  and  $\mathbf{P}^t = \{\mathbf{p}_i^t\}_{i=1}^n$ .)

### Particular and Global Best

The  $i^{\text{th}}$  *particular best* candidate at the step  $t$  is obtained via  $\mathbf{p}_i^t = \operatorname{argmin}_{\mathbf{x} \in \mathbf{N}_i^t} \{f(\mathbf{x})\}$ .

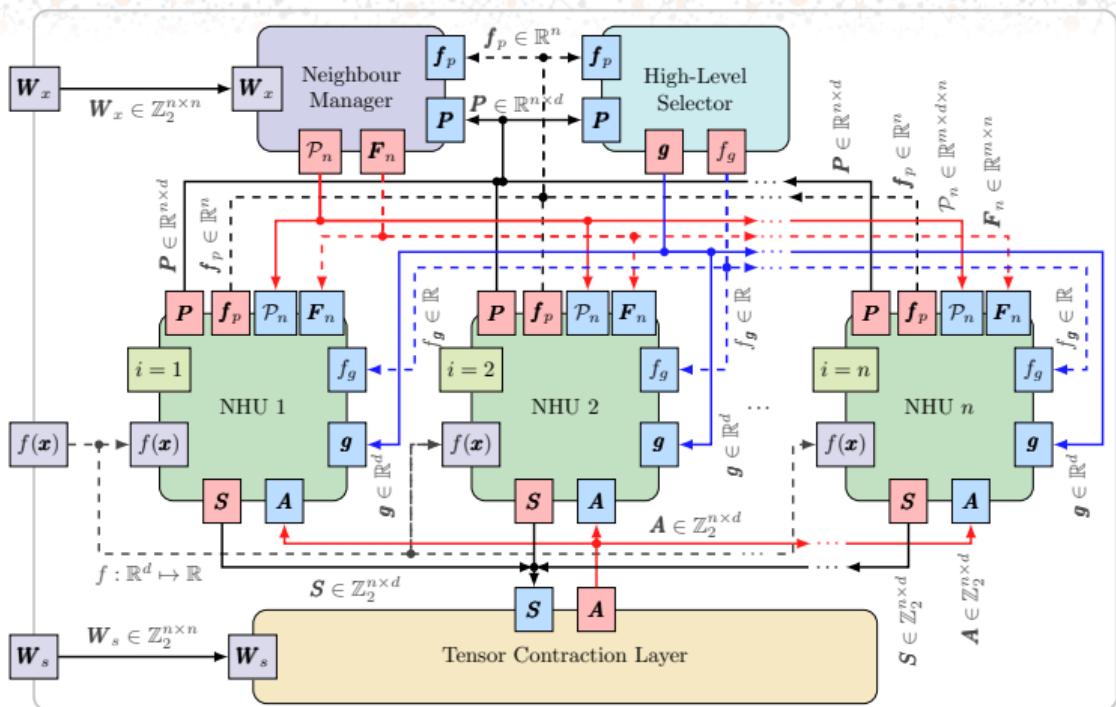
The *global best* candidate solution at step  $t$  is defined as  $\mathbf{g}^t = \operatorname{argmin}_{\mathbf{x} \in \mathbf{P}^t} \{f(\mathbf{x})\}$ .



# Proposed Framework: Neuromorphic Optimiser

The NeurOptimiser comprises:

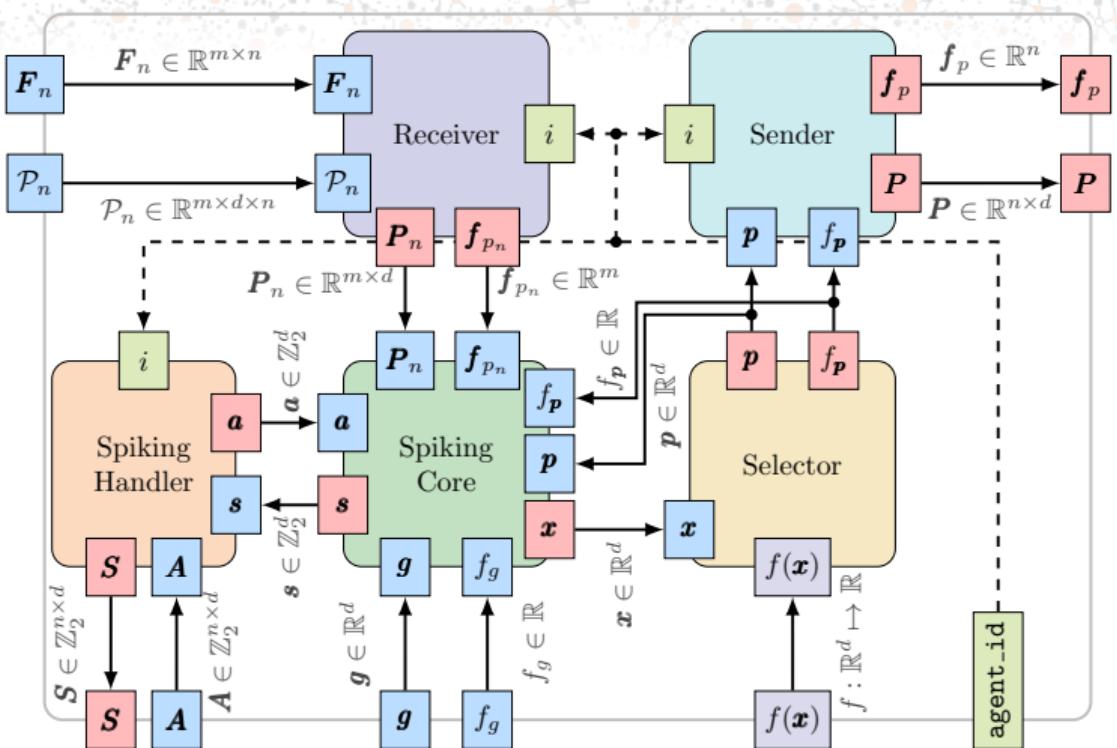
- a Tensor Contraction Layer,
- a Neighbourhood Manager,
- a High-Level Selector, and
- $n$  Neuromorphic Heuristic Units (NHUs),





## The NHU comprises:

- a Spiking Handler,
  - a Receiver, a Sender,
  - a Selector, and
  - a Spiking Core.





# Proposed Framework: Spiking Core - Overview

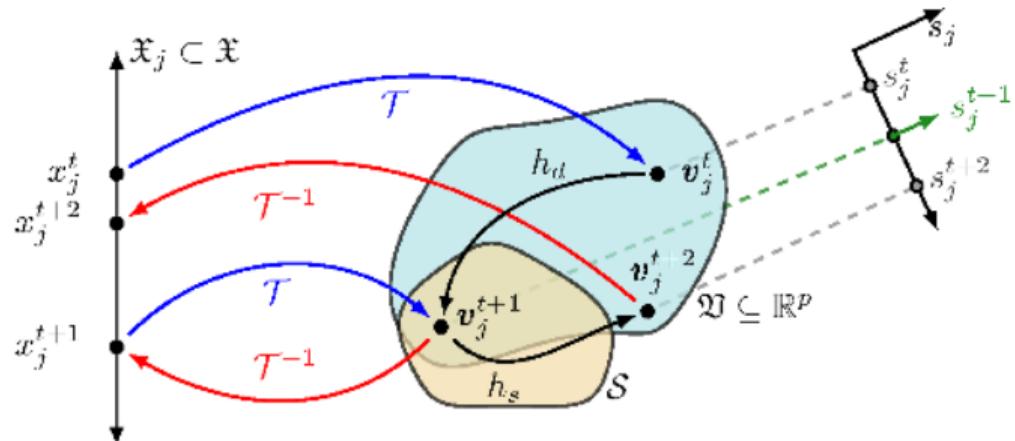


Figure: Sketch of three representative search steps performed by a spiking neuron.

## Variables

- $x_j^t \in \mathfrak{X}_j \subset \mathfrak{X}$ : Candidate solution
- $\mathbf{v}_j^t \in \mathfrak{V}$ : Neuromorphic state
- $s_j^t$ : Spike signal

## State evolution

- $\mathcal{T}$ : Bidirectional transformation
- $\mathcal{S}$ : Spiking condition set
- $h_d$ : Dynamic updating rule
- $h_s$ : Spiking-triggered rule



# Proposed Framework: Spiking Core - Key Components I

## Bidirectional Transformation $\mathcal{T}$

This is defined as  $\mathcal{T} : \mathfrak{X}_j \leftrightarrow \mathfrak{V}$ . We implemented  $\mathcal{T}$

$$\mathbf{v}_j = \mathcal{T}(x_j) \triangleq \left( \alpha(x_j - x_{\text{ref},j}), \quad 2r - (1 - x_{\text{ref},j}) \right)^T,$$

where is a reference point  $x_{\text{ref},j} \in \mathbf{x}_{\text{ref}} \in \mathfrak{X}$ , a gain factor  $\alpha \in \mathbb{R}$ , and  $r \sim \mathcal{U}(0, 1)$ .

- We considered five distinct reference modes for defining  $\mathbf{x}_{\text{ref}}$ : **null** ( $\mathbf{0}_d$ ), **p** ( $\mathbf{p}$ ), **g** ( $\mathbf{g}$ ), **pg** (midpoint)<sup>1</sup>, and **pgn** (midpoint extended with neighbours).

<sup>1</sup>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 **14**, 267–291 (2023).



# Proposed Framework: Spiking Core - Key Components II

## Spiking Condition Set $\mathcal{S}$

This is defined through a characteristic function  $\varphi_{\mathcal{S}}(\mathbf{v}_j) : \mathfrak{V} \mapsto \mathbb{Z}_2$ , such that

$$\mathcal{S} = \left\{ \mathbf{v}_j \in \mathfrak{V} \mid \varphi_{\mathcal{S}}(\mathbf{v}_j) = 1 \right\}.$$

- We implemented four spiking modes (`spk_cond`):
  - **fixed** uses a membrane potential threshold,  $\varphi_{\mathcal{S}}(\mathbf{v}_j) \triangleq |\mathbf{v}_{j,1}| > \vartheta_j^1$ ;
  - **11** and **12** that generalise it,  $\|\mathbf{v}_j\|_1 > \vartheta_j$  and  $\|\mathbf{v}_j\|_2 > \vartheta_j$ ; and
  - **stable** for shrinking radius around the origin,  $\|\mathbf{v}_j\|_2 < \varepsilon_{\text{spk}} / (1 + t)$ .
- Considering the threshold parameter  $\vartheta_j$  via distinct modes:
  - **fixed** mode considers the threshold constant  $\vartheta_j \triangleq \alpha_{\text{thr}}$ , where  $\alpha_{\text{thr}} > 0$ ;
  - **diff\_pg** strategy uses the global and particular best positions as  $\vartheta_j \triangleq \alpha_{\text{thr}} |g_j - p_j|$ ; and
  - **diff\_pref** mode defines it w.r.t. the reference position as  $\vartheta_j \triangleq \alpha_{\text{thr}} |x_{\text{ref},j} - p_j|$

<sup>1</sup>E. M. Izhikevich, "Simple model of spiking neurons", IEEE Transactions on Neural Networks **14**, 1569 (2003).



# Proposed Framework: Spiking Core - Key Components III

## State variable $\mathbf{v}_j^t$ evolution

These govern the evolution of the state variable  $\mathbf{v}_j^t$ , which depends on whether the neuron is intrinsically spiking  $s_j^t$  or externally activated  $a_j^t$  by a neighbouring unit, such as,

$$\mathbf{v}_j^{t+1} \leftarrow \begin{cases} h_d(\mathbf{v}_j^t), & \text{if } s_j^t \vee a_j^t \neq 1, \\ h_s(\mathbf{v}_j^t), & \text{if } s_j^t \vee a_j^t = 1, \end{cases} \quad \text{and} \quad s_j^{t+1} \leftarrow \varphi_S(\mathbf{v}_j^t),$$

- The **dynamic updating rule**  $h_d$  is defined by the evolution of a dynamic system. If continuous,  $\dot{\mathbf{v}}_j = g(\mathbf{v}_j)$ , an approximation method must be used.
- The **spike-triggered rule**  $h_s$  modifies the state using a heuristic operator.



# Proposed Framework: Spiking Core - Key Components IV

- $h_d$  implemented from two-dimensional continuous systems with an RK4 approx.:
  - Izhikevich models<sup>1</sup>,  $\dot{\mathbf{v}}_j \triangleq \left( 0.04v_{j,1}^2 + 5v_{j,1} + 140 - v_{j,2} + I_j, \quad a_j(b_j v_{j,1} - v_{j,2}) \right)^T$ .
  - Linear Models<sup>2</sup>,  $\dot{\mathbf{v}}_j \triangleq \mathbf{A}_j \mathbf{v}_j$ .
- $h_s$  implemented with four options:
  - `random`, a random state.
  - `fixed`, the best-known state so far with low noise.
  - `directional`, a current-to-best directional state.
  - `differential` based on the mutation operators from Differential Evolution (DE)<sup>3</sup>, using the neighbourhood for sampling random states.  
For example, `DE/rand`, and `/current-to-rand`, `/best-to-rand`, and `/current-to-best`.

<sup>1</sup>E. M. Izhikevich, "Simple model of spiking neurons", IEEE Transactions on Neural Networks **14**, 1569 (2003).

<sup>2</sup>H. S. Steven, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*, (CHAPMAN & HALL CRC, 2024).

<sup>3</sup>A. W. Mohamed et al., "Differential evolution mutations: taxonomy, comparison and convergence analysis", IEEE Access **9**, 68629–68662 (2021).



# Proposed Framework: Spiking Core - Example

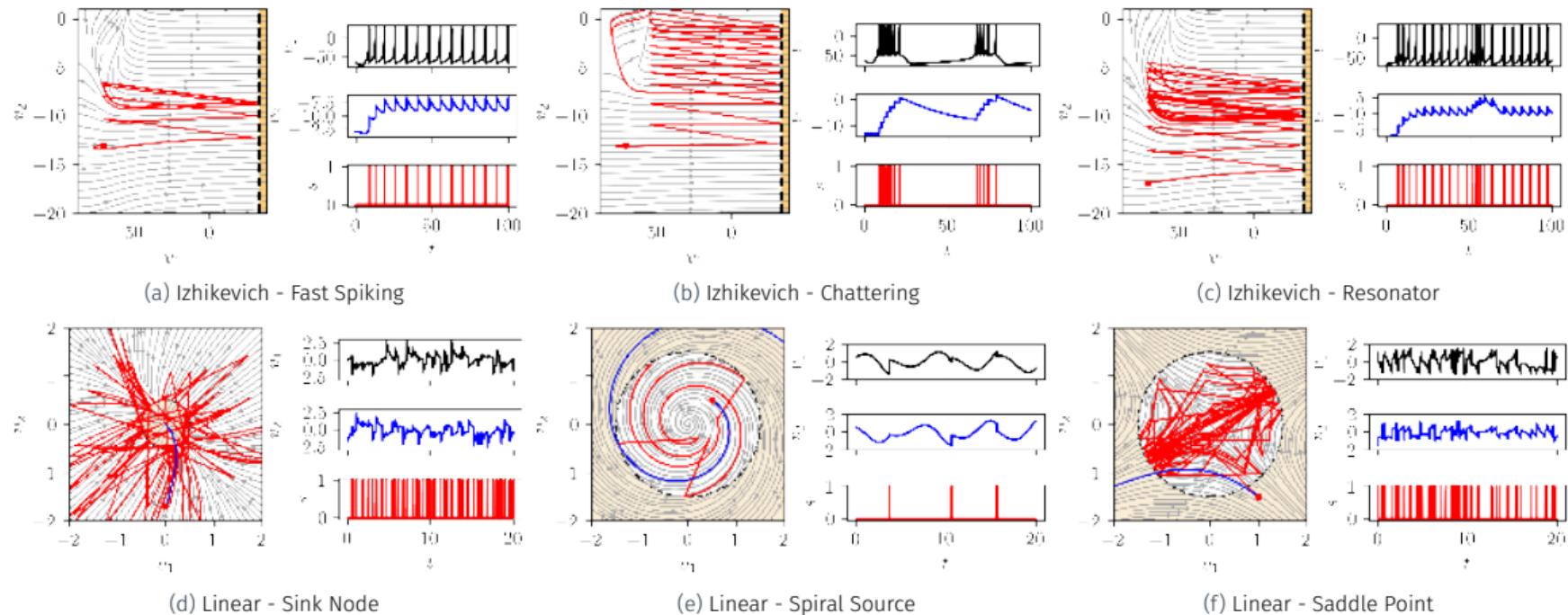


Figure: Portraits ( $v_1$  vs.  $v_2$ ) and time signals ( $v_1, v_2, s$  vs.  $t$ ) of representative neuromorphic dynamics.



# Proposed Framework: Implementation

- Our source code, including all components described in this work, is publicly available as NeurOptimiser v1.0.0 at <https://neuroptimiser.github.io>.
- All scripts, configuration files, and experimental results are freely available in<sup>1</sup>.
- NeurOptimiser was implemented in Python v3.10 using Intel's Lava framework<sup>2</sup>, which targets to Loihi devices.
- We used the noiseless BBOB test suite from the COCO platform, comprising 24 standardised problems<sup>3</sup> to evaluate our proposal. We also considered IOH<sup>4</sup> to show the versatility of our approach.

---

<sup>1</sup>J. M. Cruz-Duarte and E.-G. Talbi, *NeurOptimiser: A General Framework for Neuromorphic Optimisation - Experiment Codes and Dataset*, version 1.0.0 (Zenodo, June 2025).

<sup>2</sup>Intel Neuromorphic Computing Lab, *Lava: a software framework for neuromorphic computing*, <https://github.com/lava-nc>, Accessed: 2025-03-28, 2023.

<sup>3</sup>N. Hansen et al., "COCO: a platform for comparing continuous optimizers in a black-box setting", *Optimization Methods and Software* **36**, 114–144 (2021).

<sup>4</sup>J. de Nobel et al., "IOHExperimenter: Benchmarking Platform for Iterative Optimization Heuristics", *arXiv e-prints:2111.04077* (2021).



## Experiments and Results: Methodology

- We validated the NeurOptimiser through three sequential experiments, using 30 to 90 NHUs and employing either `linear` or `izhikevich` models for  $h_d$ .
- The simulation length ranged from 100 to  $1000 \times d$  steps, depending on the dimensionality  $d$ .
- The spike-triggered rule  $h_s$  varied across predefined operators.
- The spiking condition was set to either `fixed` or 12, and network topologies included ring and fully connected configurations.
- All experiments were conducted on a 48-core AMD EPYC 7642 CPU with 512 GB RAM, under Grid'5000<sup>1</sup> using Debian 5.10 x86-64 GNU/Linux.

<sup>1</sup>Grid'5000 is a testbed supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities, as well as other organisations. For further information, visit <https://www.grid5000.fr>.



# Preliminary Experiment: Description

- **Objective:**

To test operational correctness using IOH with five 2D problems ( $f_1, f_6, f_{10}, f_{15}$ , and  $f_{20}$ ), one per function class.

- **Details:**

- Each configuration used 30 NHUs for 1000 steps.
- Two dynamic models for  $h_d$ : linear or izhikevich
- Four heuristic operators as  $h_s$ : fixed, random, directional, and DE/rand.
- The spike condition was set to fixed.
- Both topologies ( $\mathbf{W}_s$  &  $\mathbf{W}_x$ ) were defined as one-directional rings.



# Preliminary Experiment: Absolute Error

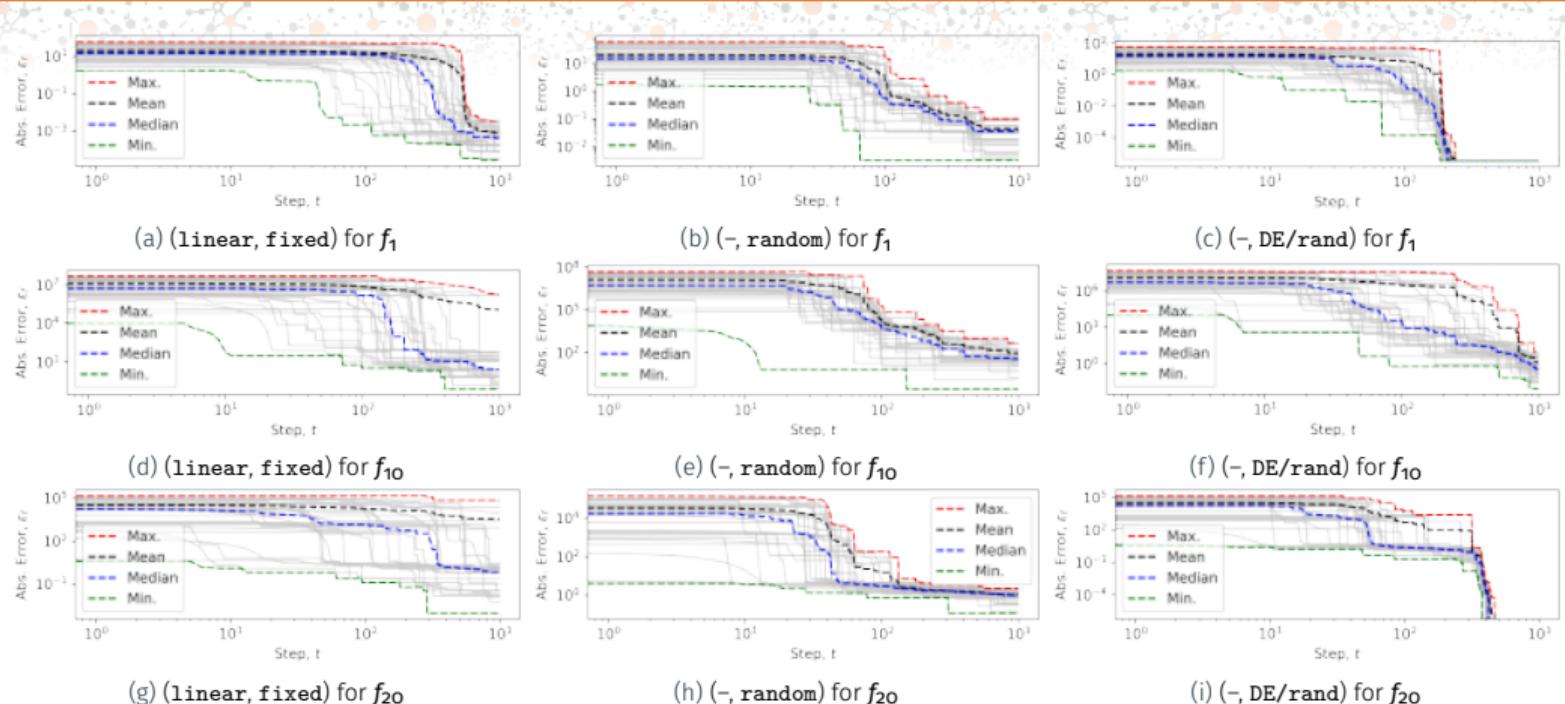


Figure: Absolute error  $\epsilon_f = |f(p_i) - f_*|$  over 1000 steps for three (of five) 2D BBOB functions, each representing a different subgroup. Each row corresponds to a function, and each column to a  $h_s$ .



# Preliminary Experiment: Positions

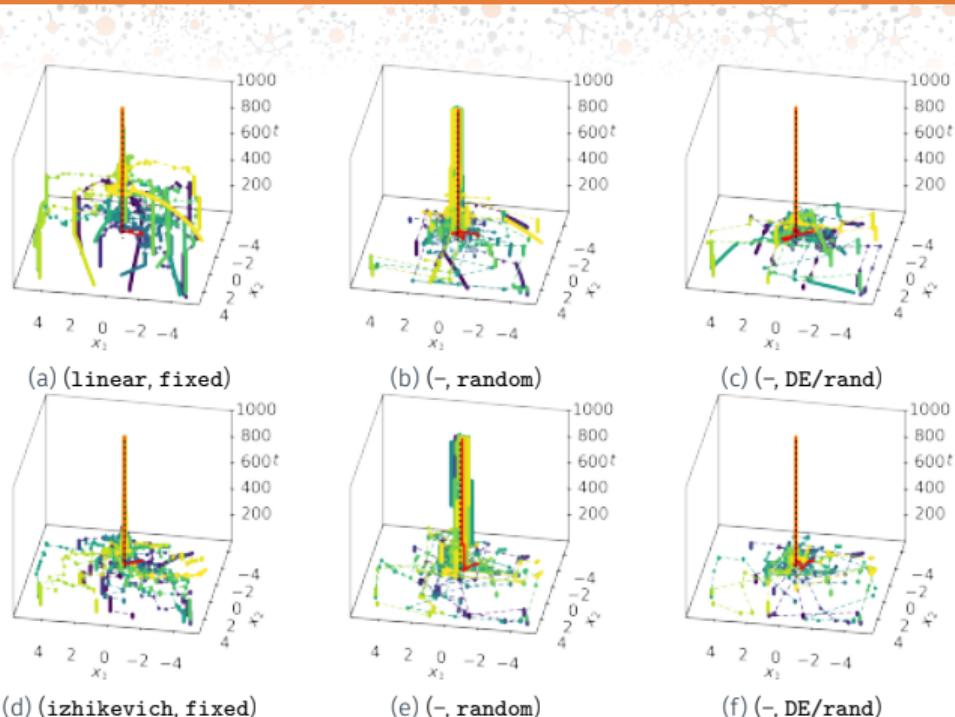


Figure: Best particular positions ( $x_1, x_2 \in \mathbf{p}_i^t$ ) of runs with two neurOptimisers with 30 NHUs searching on a 2D Sphere ( $f_1$ ) problem domain.



# Preliminary Experiment: Spiking Trains

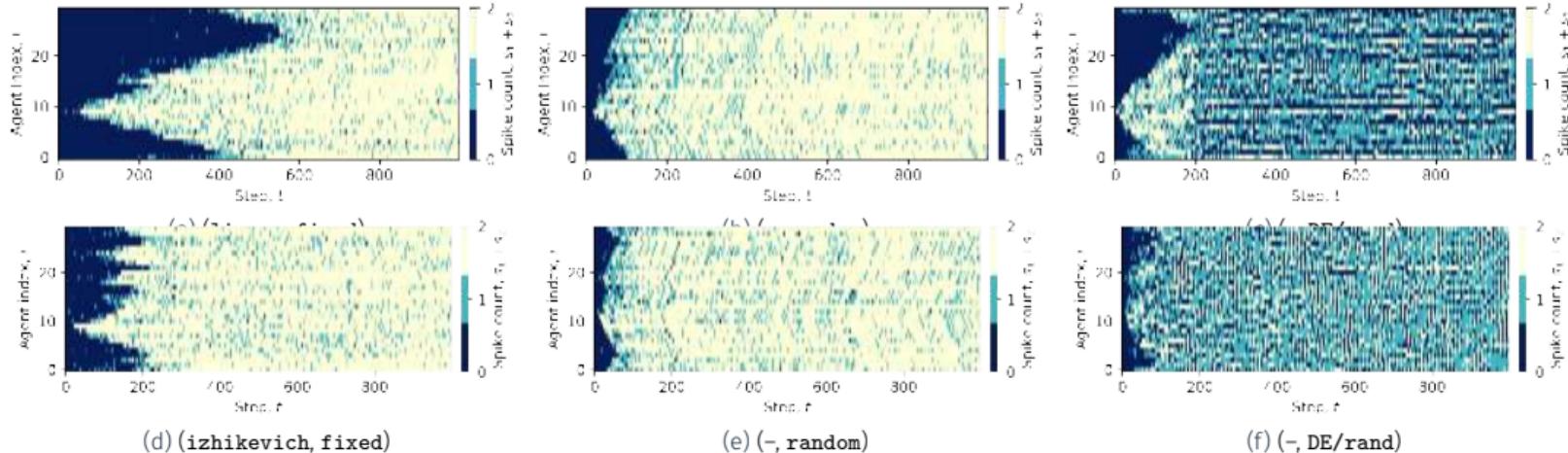


Figure: Spiking signal count ( $s_1 + s_2$ ) for each one of the 30 NHUs, implemented in neurOptimisers based on **linear** and **izhikevich** models, searching on a 2D Sphere ( $f_1$ ) problem domain.



# Confirmatory Experiment: Description

- **Objective:**

To evaluate the convergence and scalability using the full BBOB suite from COCO at dimensions 2, 3, 5, 10, 20, and 40.

- **Details:**

- Each run used 30 NHUs, either `linear` or `izhikevich` neuron models, for  $1000 \times d$  steps with the `DE/current-to-rand` operator and 12 spiking mode.
- We refer to these implementations as Neuropt-Lin and Neuropt-Izh.
- The BBOB suite contains 24 functions that are categorised as<sup>1</sup>:
  - f1-f5: separable (`separ`),
  - f6-f9: low (`lcond`) conditioning,
  - f10-f14: high (`hcond`) conditioning,
  - f15-f19: multimodal with adequate (`multi`), and
  - f20-f24: weak global structure (`muti2`).

<sup>1</sup>N. Hansen et al., "COCO: a platform for comparing continuous optimizers in a black-box setting", Optimization Methods and Software **36**, 114–144 (2021).



# Confirmatory Experiment: ECDFs - Overview

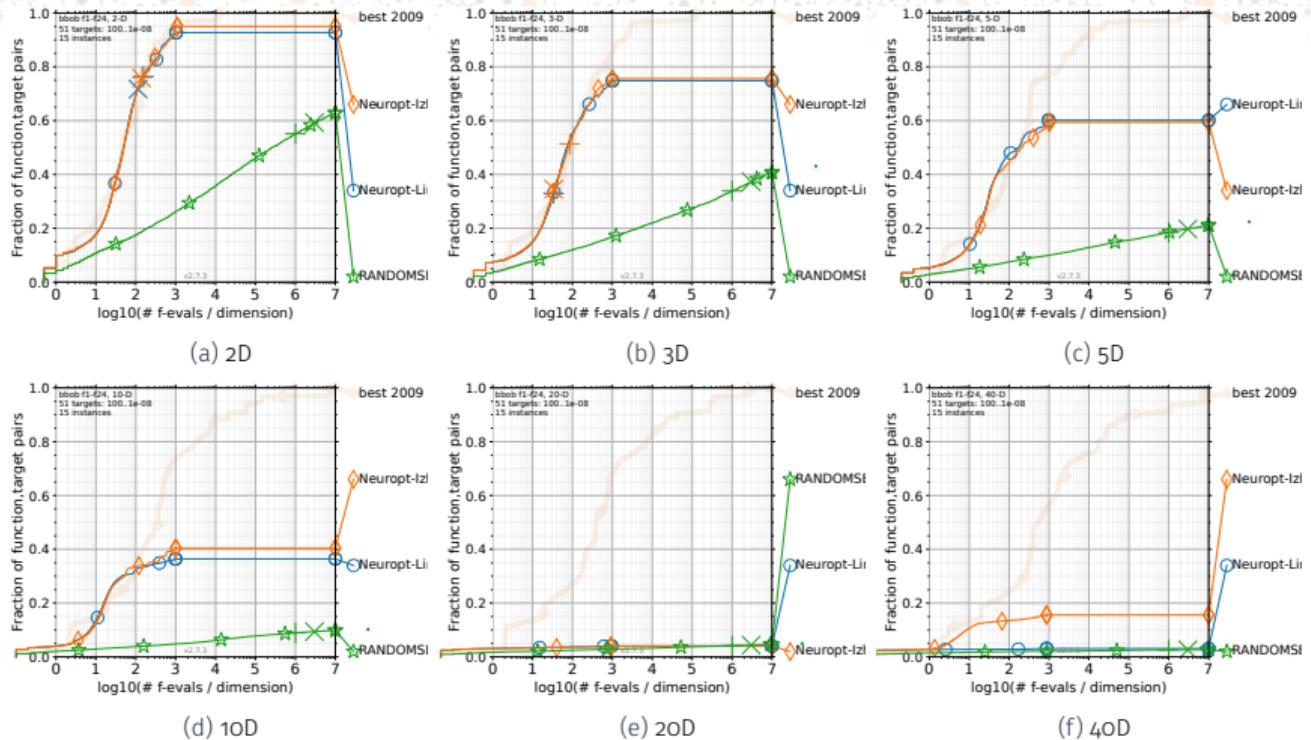


Figure: Bootstrapped Empirical Cum. Distr. Functions (ECDFs) of # f-evals/dimension for 51 targets.



# Confirmatory Experiment: ECDFs - Separable & Low Cond.

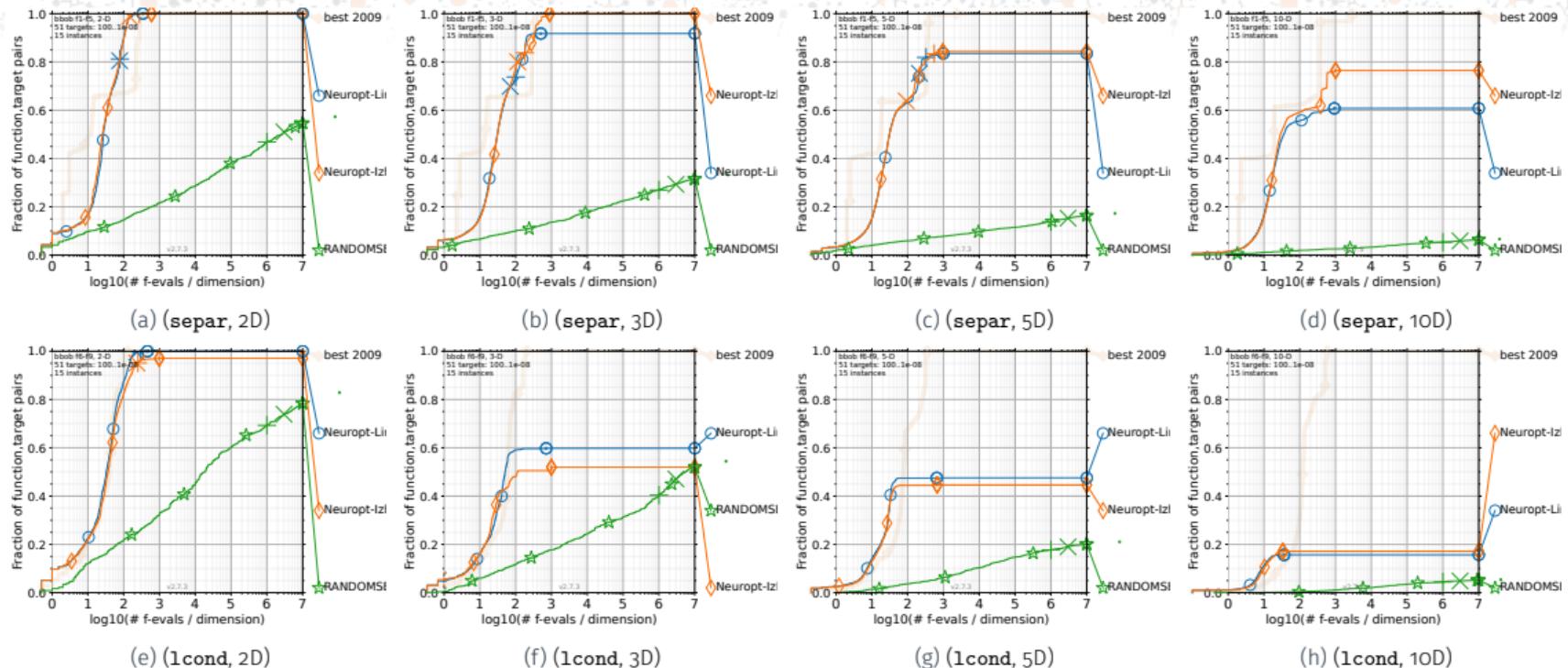


Figure: Bootstrapped Empirical Cum. Distr. Functions (ECDFs) of # f-evals/dimension for 51 targets.



# Confirmatory Experiment: ECDFs - Multi & Mult2.

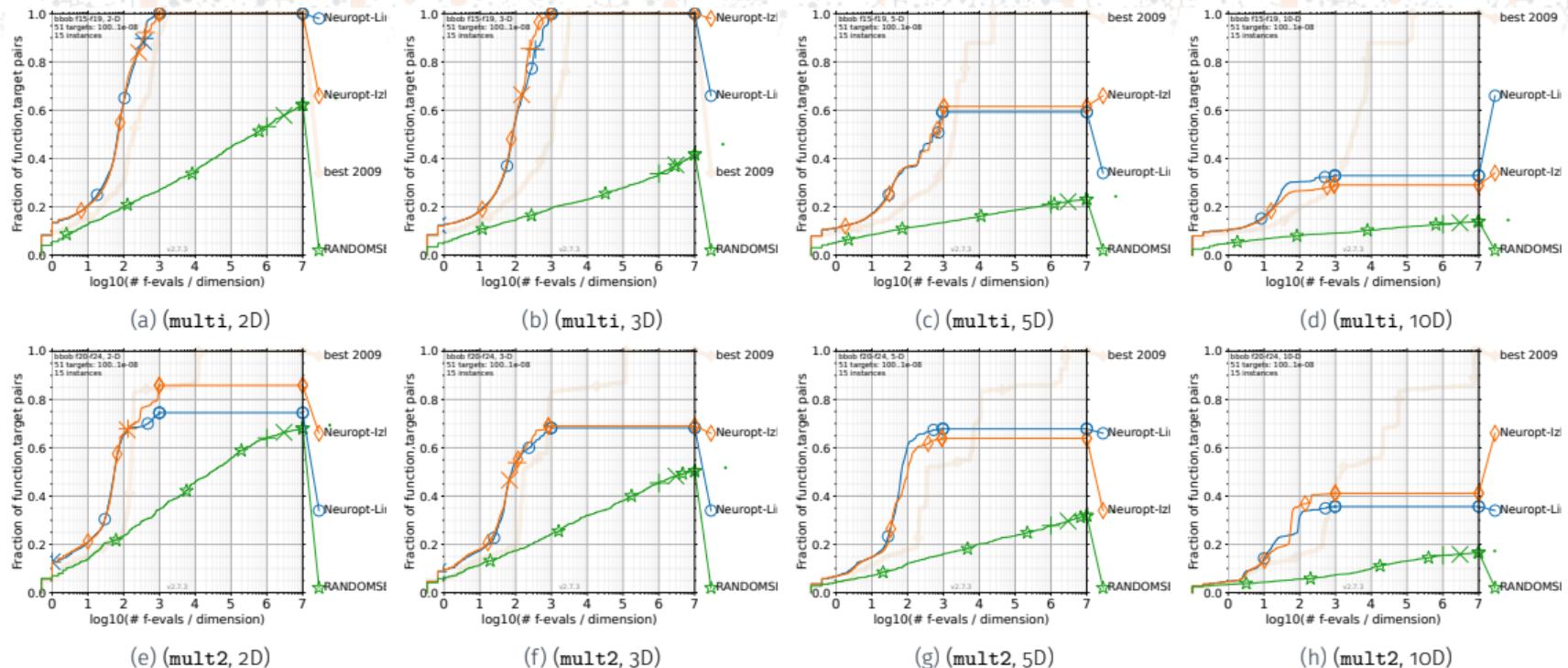


Figure: Bootstrapped Empirical Cum. Distr. Functions (ECDFs) of # f-evals/dimension for 51 targets.



# Time and Power Analysis: Description

- **Objective:**

To analyse runtime per step as well as to estimate an upper bound on the power consumption.

- **Details:**

- We increased dimensionality ( $d \in \{2, 10, 20, 40\}$ ) and population size ( $n \in \{30, 60, 90\}$ ).
- All configurations used a 50/50 split of `linear` and `izhikevich` neurons in 100-step simulations, initialised with `random_all`, and fully connected topologies.



# Time Analysis

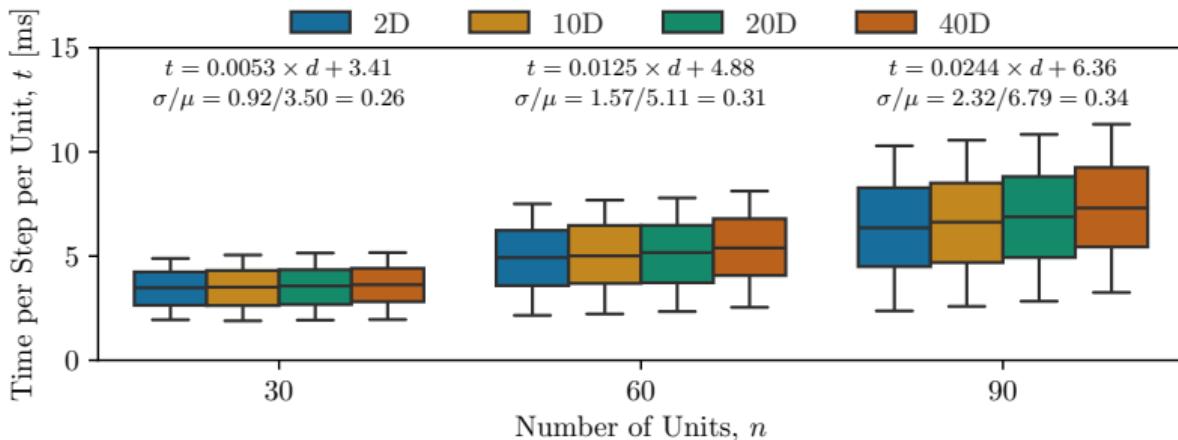


Figure: Average runtime per step and NHU ( $t$  [ms]) as a function of the number of units ( $n \in \{30, 60, 90\}$ ) for four dimensionalities ( $d \in \{2, 10, 20, 40\}$ ).



# Power Analysis

- We estimated an upper bound on the **NeurOptimiser's power consumption** when eventually executed on an Intel's Loihi device<sup>1</sup>.
- $E_{\text{step}} = 23.6 \times n_{\text{syn}} + 89.7n$  [J].
- $P_{\text{avg}} = E_{\text{step}}/\Delta t$  [W], using  $\Delta t = 0.5$  ms.
- Assuming  $n$  NHUs, dimensionality  $d$ , and  $m$  perturbations per unit per step, the number of synaptic events is  $n_{\text{syn}} = n(n - 1)md$ .
- For  $n = 90$ ,  $d = 40$ , and  $m = 89$ , we obtain  $E_{\text{step}} \approx 56.7 \mu\text{J}$ , hence  $P_{\text{avg}} \approx 1.3$  W.
- In practice, spike activity is sparse, suggesting lower energy use.

<sup>1</sup>M. Davies et al., "Loihi: a neuromorphic manycore processor with on-chip learning", IEEE Micro **38**, 82–99 (2018).



# Conclusions

- We presented a framework, **NeurOptimiser**, which embeds **optimisation logic within distributed neuromorphic units**.
- These units combine a **spiking core** with a **modular heuristic mechanism**.
- We validated this through consistent convergence, structured spiking behaviour, and competitive performance on the BBOB suite across multiple dimensions.
- The framework proved **versatile** in operating with both linear and Izhikevich dynamics.
- It revealed how spike-triggered operators, such as DE/current-to-rand, enable emergent, self-organised exploration without global control.
- We also found that its **computational cost remains manageable under CPU-based simulation**, suggesting favourable conditions for low-power neuromorphic deployment.



# Future Work

- A possible bottleneck is the communication of floating-point accuracy, which is still being developed for on-device implementations. An exciting challenge!
- To investigate adaptive topologies, reward-modulated spiking rules, dynamic selection of neuron models, and full deployment on NC hardware such as Loihi 2.
- To explore different heuristic-based rules for either  $h_d$  or  $h_s$  and meta-procedures based on the automated algorithm design, configuration, and selection field.
- This approach offers a well-supported, event-driven alternative to conventional heuristic based optimisation. The potential of this line of research is substantial!



# Complementary Information

<https://neuroptimiser.github.io/>

The screenshot shows the Neuroptimiser documentation website. At the top, there's a navigation bar with links like "Home", "Documentation", "Experiments", "Results", "Code", and "Contact". Below the navigation is a search bar. The main content area has a large image of a hedgehog-like character wearing a graduation cap, with the text "Neuroptimiser" below it. A sidebar on the left lists categories such as "Documentation", "Examples", "API", "Experiments", "Code", and "Contact". The main content includes sections like "Documentation Overview" (with a link to "https://arxiv.org/abs/2507.08320") and "Getting Started" (with steps 1-4).

<https://arxiv.org/abs/2507.08320>

The screenshot shows the arXiv preprint page for "NeurOptimisation: THE SPIKING WAY TO EVOLVE" by Jorge M. Cruz-Duarte and El-Ghazali Talbi. The page includes the abstract, authors, and publication date (July 14, 2025). The abstract discusses the framework's ability to solve complex optimisation problems using spiking neural networks. The page also features a "PDF" button and a "View on GitHub" button.



WBO 2025

INTERNATIONAL WORKSHOP ON  
BIG OPTIMISATION

Thanks!  
Grazie!  
Merci!  
Multmesc!  
¡Gracias!



Contact information

Jorge M. Cruz-Duarte

jorge.cruz-duarte@univ-lille.fr

El-Ghazali Talbi

el-ghazali.talbi@univ-lille.fr

→<https://neurooptimiser.github.io>

 Université  
de Lille

 cnrs

 inria

 CRIStAL  
Centre de Recherche en Informatique  
Signal et Algorithmique de Lille