SENMap: Multi-objective dataflow mapping & synthesis for hybrid scalable neuromorphic systems

Nembhani, P. V.; Rhodes, O.; Tang, G.; Dobrita, A. F.; Xu, Y.; Vadivel, K.; Shidqi, K.; Detterer, P.; Konijnenburg, M.; Schaik, G. -J. van

DOI
10.1109/IJCNN64981.2025.11227766

Publication date
2025

Document Version
Final published version

Published in
International Joint Conference on Neural Networks, IJCNN 2025 - Proceedings

SENMap: Multi-objective dataflow mapping & synthesis for hybrid scalable neuromorphic systems

Prithvish V Nembhani*†‡, Oliver Rhodes*, Guangzhi Tang§
Alexandra F Dobrita†, Yingfu Xu†, Kanishkan Vadivel†, Kevin Shidqi†, Paul Detterer†
Mario Konijnenburg†, Gert-Jan van Schaik†, Manolis Sifalakis†, Zaid Al-Ars‡, Amirreza Yousefzadeh¶
*University of Manchester-Advanced Processor Technologies, United Kingdom, †imec-NL, Netherlands,
‡Delft University of Technology, Netherlands, §Mastritch University-DACS, Netherlands, ¶University of Twente, Netherlands

2025 International Joint Conference on Neural Networks (IJCNN) | 979-8-3315-1042-8/25/$31.00 ©2025 Crown | DOI: 10.1109/IJCNN64981.2025.11227766

***Abstract*—This paper introduces SENMap, a mapping and synthesis tool for a scalable energy efficient neuromorphic computing architecture frameworks. SENECA a flexible architectural design optimized for executing edge AI SNN/ANN inference applications efficiently. To speed up the silicon tapeout and chip design for SENECA, an accurate emulator SENSIM was designed. While SENSIM supports direct mapping of SNNs on neuromorphic architectures, as the SNN/ANN grow in size, achieving optimal mapping for objectives like energy, throughput, area, and accuracy becomes challenging. This paper introduces SENMap, flexible mapping software for efficiently mapping large SNN/ANN applications onto adaptable architectures. SENMap considers architectural, pretrained SNN/ANN realistic examples, and event rate-based parameters and is open-sourced along with SENSIM to aid flexible neuromorphic chip design before fabrication. Experimental results show SENMap enables 40 percent energy improvements for a baseline SENSIM operating on timestep asynchronous mode of operation. SENMap is designed in such a way that it facilitates mapping large spiking neural networks for future modifications as well.** [1]

***Index Terms*—SNN/ANN mapping, SENECA, SENSIM, SNN/DNN inference mapping, large scale, accuracy, scalability, energy efficiency, closed-loop synthesis, data-flow.**

I. Introduction

Recent advances in artificial intelligence (AI) have allowed machines to perform complex tasks traditionally reserved for human intelligence. Deep neural networks (DNNs) have achieved remarkable results in image recognition, natural language processing, and reinforcement learning. As AI tasks become more complex, neuroscience insights are increasingly applied to enhance AI systems. Neuromorphic computing, inspired by the human brain, aims to mimic neural principles in AI architectures. Unlike traditional AI, which relies on discrete computations, neuromorphic systems emulate spiking neuron behavior and temporal dynamics, offering advantages such as improved energy efficiency, robustness, and real-time processing.

Neuromorphic computing, a form of bio-inspired computing, replicates the brain's capabilities for tasks such as audio/video processing and decision-making [1]–[6]. Advances include silicon-based implementations and software emulators such as Lava SDK [7], PyCARL [8], and Nengo [9]. Spiking Neural Networks (SNNs), central to neuromorphic computing, use spike-based communication to mimic neuronal activity, improving energy efficiency and biological plausibility [3], [10]–[13]. Neuromorphic systems use specialized cores, such as Loihi crossbars or ARM cores on SpiNNaker, to handle neuron dynamics and synaptic weights. Software tools such as sPyNNaker [?], [14], PyCARL [8], SpiNeMap [15], and DFSynthesizer [16] optimize SNN mapping. However, scalability and adaptability challenges remain. As larger-scale SNN applications develop [17]–[20], addressing these issues is crucial.

To address scalability, accuracy, and adaptability challenges while improving power/energy efficiency and latency, we introduce SENMap. This paper details SENMap, an extension of SENSIM [21], which enhances SNN mapping and synthesis, focusing on mapping, design space exploration, and closed-loop synthesis for practical applications on SENECA. SENMap is adaptable to various neuromorphic designs. Section II gives a brief description of SENECA [22]–[27] and SENSIM, and Section III discusses parallel developments to SENMap and compares SENMap with other neuromorphic mapping tools. Section IV describes the design of SENMap and Section V presents the details of the experimentation and results. Section VII concludes the paper summarizing contributions and future work.

II. SENECA & SENSIM Brief

SENECA [23]–[27] is a RISC-V-based digital neuromorphic processor designed for SNN/ANN computations in edge applications with limited energy. It handles unstructured spatiotemporal sparsity and irregular data transfers with its optimized neuromorphic coprocessor, three-level memory hierarchy, and dual-controlling system. The processor supports data resolutions of 4b, 8b, and Brain-float16, facilitating advanced learning and optimization. SENECA features an event-based NoC that supports multicasting, source-based routing, and data compression. Its configuration is customizable, including the number of cores, the neural processing elements (NPEs) per core, and optional off-chip memory. SENSIM [21], an open source event-based simulator developed in Python, helps simulate large SNN/ANN models with hardware-aware parameters, providing estimates of energy and latency. It models systems with data transfer between processors in different

[1]Source code of the mapper can be found here. https://github.com/Prithvish04/senMap_paper_submission & https://github.com/Prithvish04/senMap_experiments

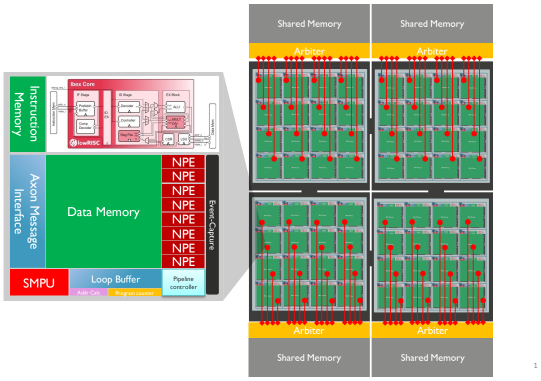

Fig. 1: A 64-core SENECA architecture [22]

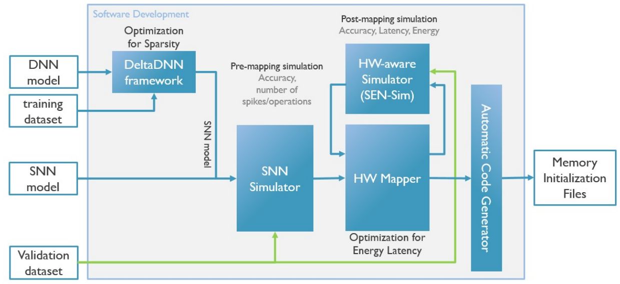

Fig. 2: SENECA SDK [22]

clock domains and includes parallel processing support for improved execution time. Unlike other simulators such as PyCarl [8], Carlsim [28], and Noxim++ [15], SENSIM offers detailed temporal energy and latency measurements based on hybrid events and clocks and supports a flexible simulation environment. Fig.1 shows an example of SENECA with 64 compute clusters, each comprising an Ibex RISC-V core, memory units, Axon Message Interface (AMI), NoC, Shared Memory Prefetch Unit (SMPU), and NPEs. Fig.2 shows the SENECA Software Development Kit (SDK), which includes the DeltaDNN framework, the SNN simulator, the hardware-aware simulator (SENSIM), the SENECA mapper (SENMap) and the code generator. The hardware details for SENECA are detailed in the prior work [27]. The energy, communication, timing, architecture and simulation parameters extracted from lower-level hardware measurements and used for the experiments are mentioned in the prior work [21].

III. LITERATURE REVIEW

In addition to SENMap, various research efforts have focused on mapping spiking neural networks (SNNs) to neuromorphic systems. The compiler for Loihi [29] efficiently maps neurons to Loihi's cores using a power-efficient greedy algorithm tailored to its crossbar architecture. SpiNeMap [15] organizes SNNs into clusters to minimize spike counts and optimize placement through Particle Swarm Optimization, differing from SENECA's multicast-based NoC architecture, which employs strategies include interspike distortion and energy-efficient hop reduction. SNEAP [30] uses multilevel graph partitioning to minimize NoC hop counts and enhance energy efficiency, while Spinnaker2 [31] proposes a similar partitioning strategy to SENECA's. However, SENMap advances this by incorporating additional architectural parameters. The method in [32] introduces a greedy clustering approach with hill climbing for real-time SNN remapping, though its application to SENECA's near memory design presents challenges. The approach in [33] partitions SNNs to optimize performance but does not address energy consumption, unlike DFSynthesizer [16], which enhances throughput and energy efficiency through data-flow based scheduling. ALPINE [34] maps various neural network topologies onto crossbar architectures but lacks validation with realistic SNNs, a gap addressed by SENMap through diverse application benchmarking. eSpine [35] focuses on prolonging memristor lifespan by partitioning workloads and optimizing mapping, contrasting with SENMap's event-based NoC approach. NeuMap [36] simplifies the mapping process using meta-heuristics and demonstrates significant improvements over previous methods, but focuses on crossbar architectures without considering architectural parameters. The method proposed in [37] uses Hilbert curves and force directed algorithms for large-scale SNNs but does not incorporate architectural design space exploration. R-MaS3N [38] improves fault tolerance through neuron reuse and heuristic partitioning, offering scalable solutions for neuromorphic systems. EdgeMap [39] employs multi-objective optimization to enhance performance, latency, and energy efficiency in edge computing scenarios, but SENMap scales more effectively with large neural networks and NoC architectures, estimating a broader set of metrics, including throughput and congestion. Table III compares SENMap with other parallel neuromorphic mapping solutions.

IV. SENMAP

To optimize processor mapping and configuration for various applications, SENMap (Scalable energy-efficient neuromorphic computing architecture mapper) was developed. It addresses the challenge of mapping event-based large-scale SNNs/DNNs on SENECA. SENMap mirrors the flexibility of SENECA and SENSIM, offering a range of algorithms and optimization methods, including single and multiobjective strategies. The following subsections detail SENMap's features and improvements. Fig. 4 illustrates the mapping of SNN/DNN applications in SENSIM.

1) Clustering and partitioning: SENMap offer strategies such as partitioning neurons layer-wise, channel-wise, height-wise, and width-wise in a homogeneous and greedy fashion across cores and clustering adjacent neurons layer-wise before distributing them across cores based on the prior information from the analytics framework, which facilitate flexible division of neurons and synapses. In SENSIM, combining layers on a core can lead to inter-event/spike distortion and incorrect updates timestamp, leading to erroneous outputs if not handled

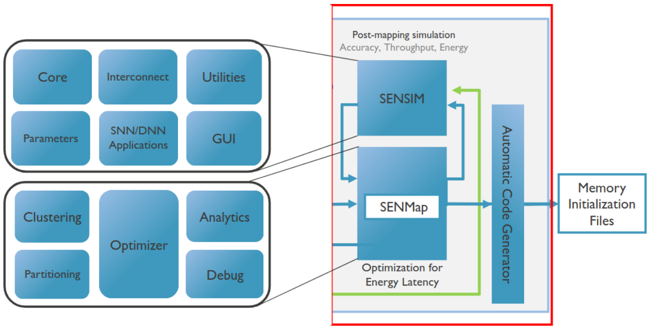

Fig. 3: SENSIM and SENMap software framework

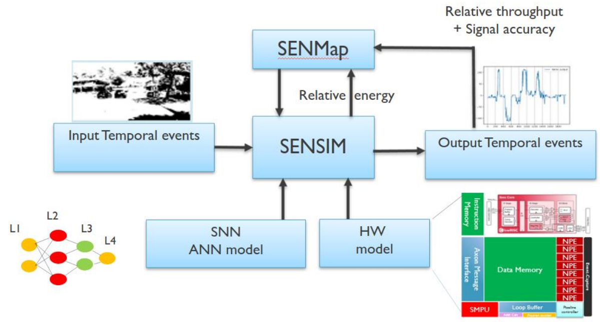

Fig. 4: Mapping of PilotNet SNN Models to SENSIM

properly. For example, clustering PilotNet-SNN layers 3 and 4 on the same core can cause deviations, as shown in Fig. 5. For smaller SNN models, inter-spike distortion is minimized by spike count but for large scale we suggest measuring accuracy estimated using end signal correlation methods as detailed in sec. IV-5.

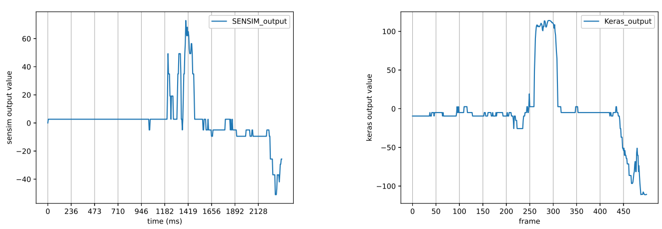

Fig. 5: Comparison of the output of SENSIM for PilotNet (500 frames) with Keras (the expected output) on clustering and partitioning layers at the same time. Outputs affected by inter-spike/inter-event distortion visible on the left.

2) Re-mapping/compression: After partitioning and clustering, the tool includes a mapping compression utility to reduce the design footprint and energy consumption by minimizing latency across SENECA cores. SENMap employs the following compression schemes:

- **Strict area optimal**: This scheme determines the mesh-size by finding factors of the total number of cores that minimize their sum. For example, with 30 cores, the factor pairs **[(1,30), (2,15), (3,10), (5,6)]** are evaluated, and the pair **(5,6)** is selected, resulting in a mesh size of **[5,6]** or **[6,5]**.
- **Loose area optimal**: For prime numbers of cores, such as 31, which only has factors **[(1,31)]**, the layout may be area inefficient. To improve flexibility, an additional core is added to make the total non-prime. For instance, with 31 cores, adding one results in 32 cores with factor pairs **[(1,32), (2,16), (4,8)]**, where **(4,8)** is chosen, giving a mesh size of **[4,8]** or **[8,4]**. [2]
- **Strict square**: This scheme aims to create a square-like mesh, even if the number of cores does not perfectly fit into a square. For example, with 26 cores, which typically results in a mesh size of **[2,13]**, the size is adjusted using a formula to approximate a square shape. mesh[rows, colums] = $\lceil\sqrt{totalcores}\rceil$

3) Bounds and constraints: [3] Lower bounds are determined by the data memory size, technology node, quantization schemes and the SNN/DNN being mapped. For example, with a 1MB node, $M_{pc} \leq 1$ MB.

$$M_{pc} = 2 \times (N_{npc} + (N_{tpc} \times F_{snn}) \times (BW_{states} + BW_{outputs})) + (N_{wpc} + N_{bpc}) \times BW_{weights} \tag{1}$$

where:

N_{npc}	neurons per core
N_{wpc}	weights per core
N_{bpc}	biases per core
N_{tpc}	thresholds per core
M_{pc}	memory per core
BW_{states}	bit-width of neuron states
$BW_{outputs}$	bit-width of outputs
$BW_{weights}$	bit-width of weights and biases
F_{snn}	flag for SNN (true if SNN)

With SENSIM, the chip architect determines the upper bound, as the total number of cores is considered infinite.

4) Optimization algorithms & problem design: SENMap utilizes single, multi-, and many-objective optimization through the Pymoo framework [40], which includes algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Nelder-Mead [41], ISRES [42], NSGA-II, NSGA-III [43], and MOEA/D [44]. Integrating Pymoo allows SENMap to leverage various optimization algorithms and parameters, which is beneficial for tackling the nonlinear problems involved in mapping & closed-loop synthesis of large neural networks. Metaheuristics in particular are the most suitable for these challenges considering the diversity of SNNs, the temporal dynamics in the system, and the evolving nature of SENECA resulting in non-linearities in the system at scale. However,hyperparameter tuning or heuristics reduce complexity and solution time. To formulate the problem for

[2]Loose area optimal mapping scheme is considered when combining different large nerual networks

[3]The details on how we use the parameters and form the bounds are mentioned in the previous paper [21]. All the experiments were carried out in asynchronous time step mode of SENSIM

TABLE I: Summary of Parallel Developments in Spiking Neural Network Mapping

Mapping	Clustering	Partitioning	Optimizing Parameters	Architecture	Network type	Key difference
Loihi compiler [29]	greedy, resource-constrained	greedy, custom graph	cross-sectional bandwidth, cut penalty	Loihi, crossbar, in-mem	small/synthetic	novel graph and cut penalty based mapper
SpiNeMap [15]	heuristics (SpiNecluster, spikecount)	metasueristics (PSO)	spike-count, communication, Inter-Spike Distortion, latency, energy	crossbar,in-mem (DYNAPSE)	small realistic FFN (MLP-MNIST, LeNet-MNIST)	ISI distortion minimization
SNEAP [30]	multilevel graph	metaheuristics (SA,PSO,Tabu)	energy, spike-count, average hop	crossbar, in-mem	small, synthetic FFN	multilevel graph
SpiNNaker2 [31]	greedy, resource-constrained	layerwise, output channels, height/weight	communication, congestion, datareuse	NoC, NearMem	realistic (ResNet, VGG16)	NearMem mapping, datareuse
Runtime [32]	greedy	hill-climbing	communicated spikes	crossbar, in-mem	small, synthetic	runtime
SNN compiler [33]	SDFG	greedy	performance, latency	crossbar, in-mem	small, realistic (LeNet, CNN, MLP, Edgdet)	SDFG, Maxplus-Algebra
DFSynthesizer [16]	homogeneous	dataflow scheduling	throughput, energy	crossbar, in-mem	small realistic (VGG16, AlexNet, LeNet)	SDFG
ALPINE [34]	semi-stochastic	Graph-ML, adjacency-list, iterative	throughput	crossbar, in-mem	small, synthetic (FFN, AE, SOM, LSM, RNN)	different SNN typology
eSpine [35]	KL graph	metaheuristics(PSO)	fault, energy, lifetime	crossbar, memristors NoC	CNN,MLP,RNN	endurance-aware
NeuMap [36]	heuristics (spike-count)	meta-heuristics (multilevel graph partitioning)	spike-firing-rate, communication patterns	crossbar, in-mem	small realistic (CNN-CIFAR, MLP-MNIST, LeNet etc)	multi-level graph partitioning
Very large [37]	homogenous	hilbert curve, force-directed	energy, latency, congestion	multicore 2D-NoC	very large realistic/synthetic (AlexNet, MobileNet, ResNet)	very large SNN
R-MaS3N/ NR-NASH [38]	analytical (spike-count, synaptic-connection)	heuristic	accuracy, communication	3D-NoC	small, synthetic, FFN (MLP)	analytical, 3D-NoC
EdgeMap [39]	custom convex function (size, spike-count)	flow-based,two-stage, multi-objective	energy, communication, throughput, Hop, latency, congestion	agnostic	small realistic (MLP-MNIST, LeNet)	relatively fast
SENMap	**homogeneous, greedy**	**meta-heuristics, multi-objective, channel, height/width**	**energy, latency, area, accuracy, architectural**	**scalable, flexible, NoC, NearMem**	**large realistic SNN/DNN (PilotNet, MLP-MNIST)**	**rate-based, architectural, closed loop, flexible**

optimization, SENMap evaluated several metaheuristics and tools, including jMetalPy [45], PyGMO [46], Metaheuristics.jl [47], Open Beagle [48], Opt4J [49], and JGAP [50]. Pymoo was the most suitable, supporting a wide range of optimization problems, including random search for binary variables, discrete variables, permutations, mixed variable types, custom variables, biased initialization, and subset selection.

5) Inter-spike/event distortion and accuracy measurement: Inter-spike distortion is a significant challenge when mapping neurons onto neuromorphic hardware [15]. While previous approaches [8], [28] focus on minimizing distortion by individually analyzing spike timing, our method assesses signal integrity by comparing the correctness of two end signals, $x(t)$ and $y(t)$, using cross-correlation to produce a normalized correlated signal $z(t)$. The approximate shift in the end signal is then estimated.

$$z(t) = \frac{\mathrm{corr}(x(t), y(t))}{\sqrt{(x(t) \cdot y(t))(x(t) \cdot y(t))}}; t_{shift} = \arg\max_{t} z(t) \tag{2}$$

6) Data collection and visual debugging: SENMap collects data for all the optimization iterations to train a model, approximate the energy and latency while running optimization experiments. The data could be used towards designing data-driven heuristics. Adaptive mapping in real time and energy reduction is a feature that is in development [51].

7) Flexible SNN/ANN replacement: Since SENSIM takes into account the mapping of several large-scale SNN and DNN with temporal delta activation convolution and dense layers hence SENMap was designed in such a way that easy selection and replacement of SNN/ANN is possible. SENMap gives the flexibility to optimize on a hybrid SNN-ANN architecture [52], [53] in the future and other topologies as suggested in [34]. Fig 6 gives an overview of details included in the data collection and debugging framework. The framework also suggests that energy in the core is accumulating linearly per core when PilotNet is mapped to it. On varying parameters the accumulation of the energy per core is piecewise linear.[4]

8) Towards closed loop synthesis: SENMap is not only designed for optimal neural-network mappings on SENECA but also supports hardware architecture and SNN/ANN event model parameters co-optimization to optimize energy, latency, area, and accuracy. It maps neuron states, weights, biases, and thresholds to the core while allowing flexibility in neuromorphic parameters like processing elements, memory, bit-width, clock frequency, and flit-width.

9) Parallel metaheuristics: Parallel meta-heuristics enhance solution quality by running multiple searches simultaneously, particularly in population-based algorithms such as PSO [54]. The Pymoo framework leverages Python's multiprocessing and Dask [55] to accelerate tasks including processing 2000 images, which can be substantially improved with parallel execution on a 30-core cluster node, requiring around ≈ 6 GB of memory per thread.

V. Experimental Results

This section presents experiments with SENMap, focusing on optimizing architectural and event rate-based parameters

[4]The SNNs used in the experiments are not trained from scratch but are rather converted from pretrained ANN networks and spatial temporal sparsity is extracted from them

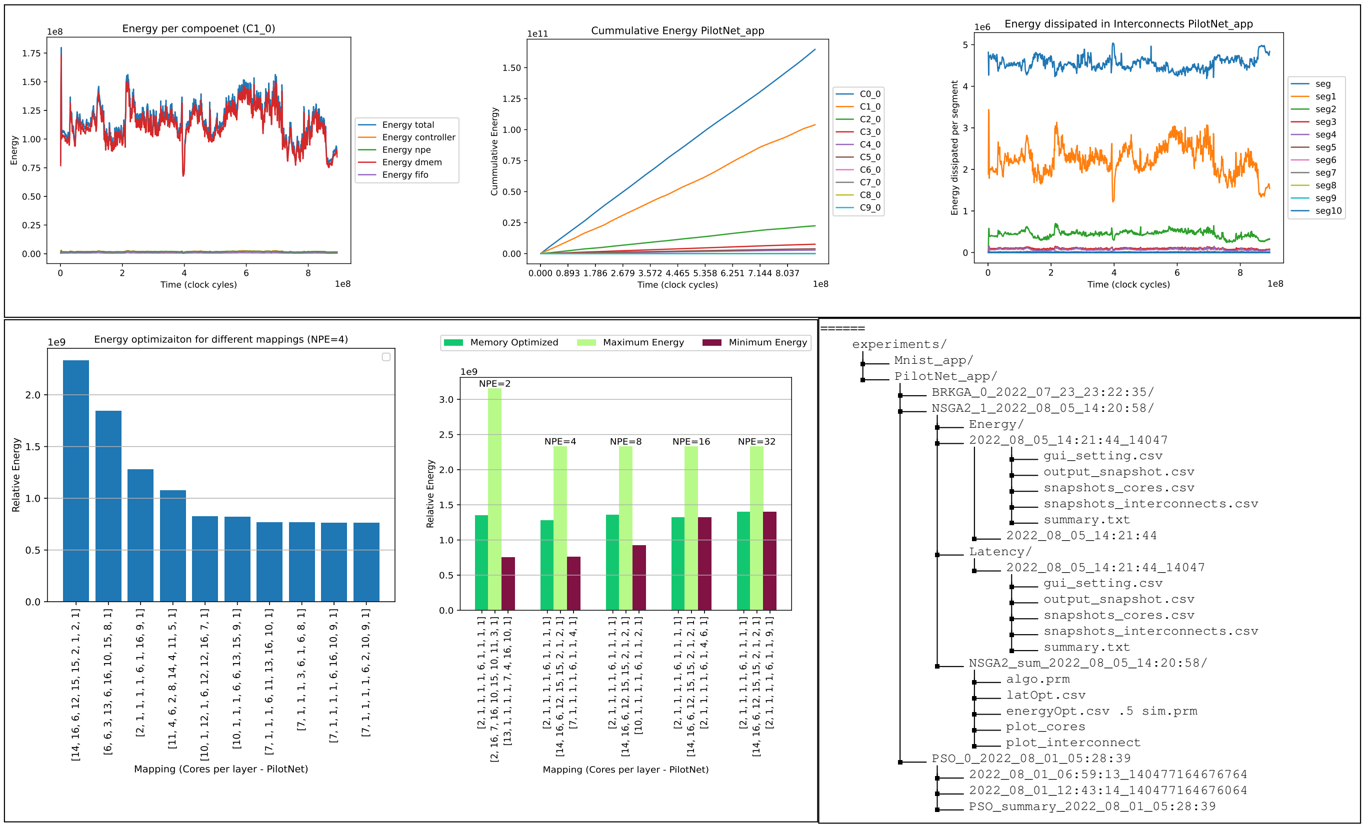

Fig. 6: Analytics and data collection framework for Pre and Post optimization

for the neural network [56] (subsequently named PilotNet). Using varied dataset sizes (#10-#30 frames), we observed that dataset size and energy efficiency minimally affected results. A mathematical model for the neuromorphic core, detailed in [21], revealed a nonlinear system due to operations like **max**, necessitating adaptable meta-heuristics over fixed heuristics for dynamic demands in ANN and SNN applications. Despite the variety of meta-heuristics available, no single approach universally excels due to the no-free lunch theorem [57]; effectiveness depends on the problem structure, hence for experiments and synthesis we used GA and NSGA-2 with hyper-parameters mentioned in Table II, as implemented in PyMOO, to balance multiple objectives like energy, latency, and accuracy.

TABLE II: Algorithm Parameters for GA and NSGA2

algo param	GA value	NSGA2 value
population size	30	40
mutation type	integer polynomial mutation (eta=3.0)	integer polynomial mutation (eta=3.0)
crossover type	integer SBX (eta=3.0)	integer SBX (eta=3.0)
sampling type	integer (random selection)	integer (random selection)
off-springs	-	10

Parallel meta-heuristics reduced time-to-solution from 3 days to 3 hours. The results indicate that frame rate impacts the precision of PilotNet inference and mapping efficiency, as illustrated in Figs. 7, 8 & 9. These figures show that increasing frame rates, particularly to 120 fps, affects mapping, making asynchronous transfer ideal for SNNs, where fewer NPEs require additional cores for SIMD arrays. We measure a 40% energy improvement from right to left in Fig. 9 as we increase the rate and an optimal mapping requiring just 1 NPE and an increase in cores. In ANNs, rates beyond a threshold can lead to signal interleaving, affecting output accuracy. SENECA's flexibility as an edge AI chip supports both SNN and ANN needs, with SENMap providing optimal mappings across applications. The correlation values for the SENSIM output signals for the PilotNet dataset as shown in Fig. 9 can also be compared here in Table III. The end signals are normalized before correlation and the time shift is estimated in millisecond after scaling up with the time-step in the simulation used.[5]

TABLE III: Signal Cross-Correlation Results

Correlation Pair	Correlation value	Time shift(ms)
corr($fps = 30, fps = 60$)	0.89	1565
corr($fps = 60, fps = 120$)	0.88	65
corr($fps = 30, fps = 120$)	0.90	1705
corr($fps = 0, fps = 120$)	1	0
corr($fps = 0, fps = 60$)	0.88	65
corr($fps = 0, fps = 30$)	0.90	1705

[5]fps=0 corresponds to a complete event driven system

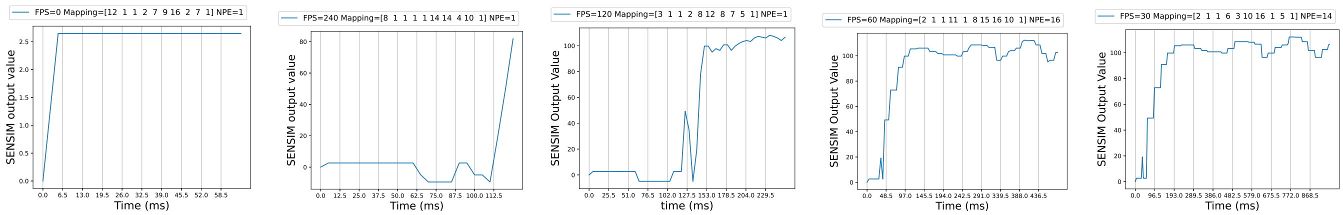

Fig. 7: SENSIM output comparison (#260-#290) frames for different mapping schemes, event/frame rate & neural processing elements

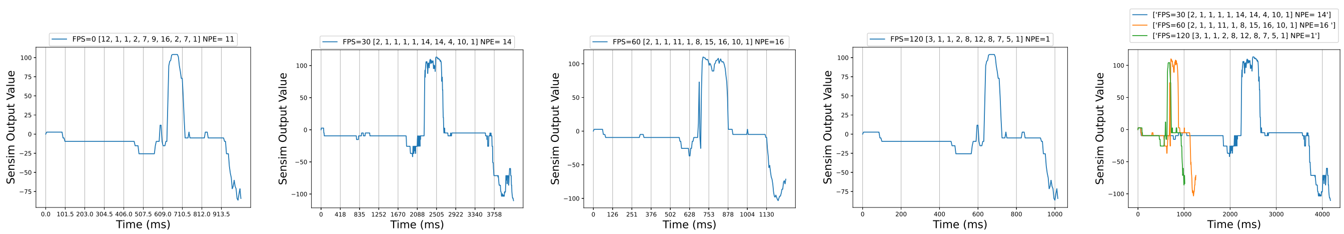

Fig. 8: Comparing the output signal of PilotNet for fps=30,60,120,0 (event-based) (left to right) for a dataset of (#0-#500) frames for different mapping & number of NPEs

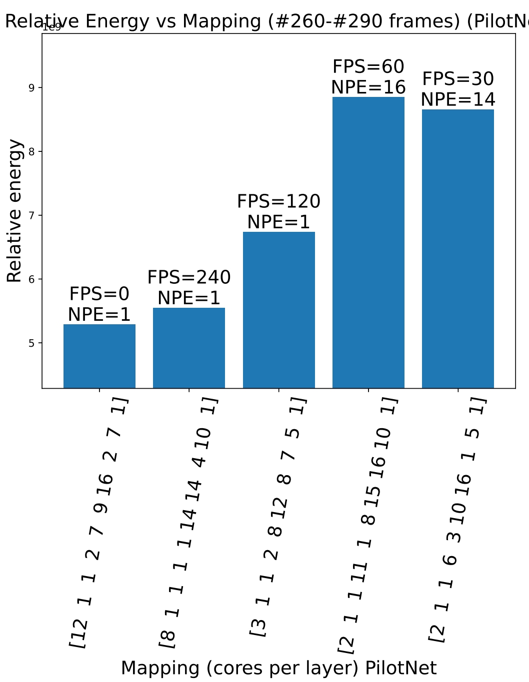

Fig. 9: Relative energy vs mapping for (#260-#290) frames (PilotNet) at different rates

VI. Discussion

SENECA over the last 4 years has undergone several changes from having SIMD NPE with loop buffer [22] emulating a GALS quantized system to double controlled system with varying levels of synchronicity in a core-to-core system [27], mapping large pretrained SNN/ANN will grow with time. Event rate as a parameter not identified in prior work is something that needs to be considered when considering the mapping into account for event based systems. From the research presented in this paper, we conclude that event-rate is an important parameter when considering mapping in scalable energy efficient neuromorphic systems. This also opens doors to questions such as what was the rate at which frames were acquired and models were trained, what systems (sensors) were used to train the these existing models.

VII. Conclusion and Future Work

This paper addressed key challenges in mapping large SNNs onto hardware platforms through SENMap, a versatile framework integrated with existing software infrastructures. We presented flexible heuristic and meta-heuristic algorithms, including genetic algorithms and particle swarm optimization, to improve neuron partitioning and mapping efficiency. Addressing inter-spike distortion and clustering challenges, our approach minimized latency and energy while managing constraints including chip area limitations for large SNN applications. SENMap achieved around 40% energy efficiency without compromising accuracy. Future work could enhance SENMap by allowing layer combinations to reduce signal distortion, utilizing 3D core stacking and 2.5D/3D memory configurations [58] to improve large-scale SNN mapping, and expanding synthesis to include communication and routing parameters. Multi-objective optimization could further refine chip synthesis, while scaling SENMap for larger applications such as ResNet [20] would improve real-time performance and energy efficiency, advancing algorithm-hardware-software co-optimization for hybrid SNN & ANN applications.

VIII. Acknowledgment

This work was partially funded by research and innovation projects REBECCA (KDT JU under grant agreement No. 101097224), NeuroKIT2E (KDT JU under grant agreement No. 101112268), and NimbleAI (Horizon EU under grant agreement 101070679), and fully supported by friends and family.

References

[1] S. Höppner, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, M. Stolba, F. Kelber, B. Vogginger, F. Neumärker, G. Ellguth, S. Hartmann, S. Schiefer, T. Hocker, D. Walter, G. Liu, J. Garside, S. Furber, and C. Mayr, *The SpiNNaker 2 Processing Element Architecture for Hybrid Digital Neuromorphic Computing*. Mar. 2021.

[2] C. Mayr, S. Hoeppner, and S. Furber, "SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning," Nov. 2019. Number: arXiv:1911.02385 arXiv:1911.02385 [cs].

[3] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)," *IEEE transactions on biomedical circuits and systems*, vol. 12, pp. 106–122, Feb. 2018.

[4] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," *Proceedings of the IEEE*, vol. 102, pp. 699–716, May 2014. Conference Name: Proceedings of the IEEE.

[5] F. Corradi, G. Adriaans, and S. Stuijk, "Gyro: A Digital Spiking Neural Network Architecture for Multi-Sensory Data Analytics," in *Proceedings of the 2021 Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools Proceedings*, (Budapest Hungary), pp. 9–15, ACM, Jan. 2021.

[6] J. Schemmel, S. Billaudelle, P. Dauer, and J. Weis, "Accelerated Analog Neuromorphic Computing," Mar. 2020. arXiv:2003.11996 [cond-mat, q-bio].

[7] "Lava Software Framework — Lava documentation."

[8] A. Balaji, P. Adiraju, H. J. Kashyap, A. Das, J. L. Krichmar, N. D. Dutt, and F. Catthoor, "PyCARL: A PyNN Interface for Hardware-Software Co-Simulation of Spiking Neural Network," May 2020. arXiv:2003.09696 [cs].

[9] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, 2014.

[10] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, pp. 1659–1671, Dec. 1997.

[11] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip," vol. 34, no. 10, p. 21, 2015.

[12] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, and L. Shi, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, pp. 106–111, Aug. 2019. Publisher: Nature Publishing Group.

[13] D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu, Y. Shen, and G. Pan, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Journal of Systems Architecture*, vol. 77, pp. 43–51, June 2017.

[14] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. D. Rowley, A. B. Stokes, and S. B. Furber, "sPyNNaker: A Software Package for Running PyNN Simulations on SpiNNaker," *Frontiers in Neuroscience*, vol. 12, 2018.

[15] A. Balaji, F. Catthoor, A. Das, Y. Wu, K. Huynh, F. G. Dell'Anna, G. Indiveri, J. L. Krichmar, N. D. Dutt, and S. Schaafsma, "Mapping Spiking Neural Networks to Neuromorphic Hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, pp. 76–86, Jan. 2020.

[16] S. Song, H. Chong, A. Balaji, A. Das, J. Shackleford, and N. Kandasamy, "DFSynthesizer: Dataflow-based Synthesis of Spiking Neural Networks to Neuromorphic Hardware," Aug. 2021. arXiv:2108.02023 [cs].

[17] P. O'Connor and M. Welling, "Deep Spiking Networks," Nov. 2016. arXiv:1602.08323 [cs].

[18] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going Deeper With Directly-Trained Larger Spiking Neural Networks," Dec. 2020. arXiv:2011.05280 [cs].

[19] Y. Hu, H. Tang, and G. Pan, "Spiking Deep Residual Network," June 2020. arXiv:1805.01352 [cs].

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015. arXiv:1512.03385 [cs].

[21] P. Nembhani, K. Vadivel, G. Tang, M. Tahghighi, G.-J. van Schaik, M. Sifalakis, Z. Al-Ars, and A. Yousefzadeh, "SENSIM: An Event-driven Parallel Simulator for Multi-core Neuromorphic Systems," in *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, June 2024. ISSN: 2161-4407.

[22] A. Yousefzadeh, G.-J. van Schaik, M. Tahghighi, P. Detterer, S. Traferro, M. Sifalakis, M. Hijdra, J. Stuijt, M. Hartmann, F. Catthoor, I. Ocket, and M. Konijnenburg, *SENeCA: Next generation Neuromorphic Processor for EdgeAI*. Mar. 2022.

[23] G. Tang, K. Vadivel, Y. Xu, R. Bilgic, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G.-J. van Schaik, and A. Yousefzadeh, "SENECA: building a fully digital neuromorphic processor, design trade-offs and challenges," *Frontiers in Neuroscience*, vol. 17, 2023.

[24] A. Yousefzadeh, G.-J. van Schaik, M. Tahghighi, P. Detterer, S. Traferro, M. Hijdra, J. Stuijt, F. Corradi, M. Sifalakis, and M. Konijnenburg, "SENeCA: Scalable Energy-efficient Neuromorphic Computer Architecture," p. 4.

[25] G. Tang, A. Safa, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G.-J. van Schaik, and A. Yousefzadeh, "Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design," Mar. 2023. arXiv:2303.15224 [cs, eess].

[26] A. YOUSEFZADEH, G.-J. V. SCHAIK, and E. Sifalakis, "Neurosynaptic processing circuitry," Dec. 2023.

[27] Y. Xu, K. Shidqi, G.-J. van Schaik, R. Bilgic, A. Dobrita, S. Wang, R. Meijer, P. Nembhani, C. Arjmand, P. Martinello, A. Gebregiorgis, S. Hamdioui, P. Detterer, S. Traferro, M. Konijnenburg, K. Vadivel, M. Sifalakis, G. Tang, and A. Yousefzadeh, "Optimizing event-based neural networks on digital neuromorphic architecture: a comprehensive design space exploration," *Frontiers in Neuroscience*, vol. 18, Mar. 2024. Publisher: Frontiers.

[28] M. Beyeler, K. D. Carlson, T.-S. Chou, N. Dutt, and J. L. Krichmar, "CARLsim 3: A user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2015. ISSN: 2161-4407.

[29] C.-K. Lin, A. Wild, G. N. Chinya, T.-H. Lin, M. Davies, and H. Wang, "Mapping spiking neural networks onto a manycore neuromorphic architecture," *SIGPLAN Not.*, vol. 53, pp. 78–89, June 2018.

[30] S. Li, S. Guo, L. Zhang, Z. Kang, S. Wang, W. Shi, L. Wang, and W. Xu, "SNEAP: A Fast and Efficient Toolchain for Mapping Large-Scale Spiking Neural Network onto NoC-based Neuromorphic Platform," Mar. 2020. Number: arXiv:2004.01639 arXiv:2004.01639 [cs].

[31] F. Kelber, B. Wu, B. Vogginger, J. Partzsch, C. Liu, M. Stolba, and C. Mayr, "Mapping Deep Neural Networks on SpiNNaker2," in *Proceedings of the Neuro-inspired Computational Elements Workshop*, (Heidelberg Germany), pp. 1–3, ACM, Mar. 2020.

[32] A. Balaji, T. Marty, A. Das, and F. Catthoor, "Run-time Mapping of Spiking Neural Networks to Neuromorphic Hardware," June 2020. Number: arXiv:2006.06777 arXiv:2006.06777 [cs].

[33] S. Song, A. Balaji, A. Das, N. Kandasamy, and J. Shackleford, "Compiling Spiking Neural Networks to Neuromorphic Hardware," in *The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, pp. 38–50, June 2020. arXiv:2004.03717 [cs].

[34] K. Shreya, "Mapping of Spiking Neural Network Topologies on Neuromorphic Hardware | TU Delft Repository," Aug. 2021.

[35] T. Titirsha, S. Song, A. Das, J. Krichmar, N. Dutt, N. Kandasamy, and F. Catthoor, "Endurance-Aware Mapping of Spiking Neural Networks to Neuromorphic Hardware," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, pp. 288–301, Feb. 2022. Conference Name: IEEE Transactions on Parallel and Distributed Systems.

[36] C. Xiao, J. Chen, and L. Wang, "Optimal Mapping of Spiking Neural Network to Neuromorphic Hardware for Edge-AI," *Sensors*, vol. 22, p. 7248, Jan. 2022. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

[37] O. Jin, Q. Xing, Y. Li, S. Deng, S. He, and G. Pan, "Mapping Very Large Scale Spiking Neuron Network to Neuromorphic Hardware," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, (New York, NY, USA), pp. 419–432, Association for Computing Machinery, Mar. 2023.

[38] W. Y. Yerima, K. N. Dang, and A. B. Abdallah, "R-MaS3N: Robust Mapping of Spiking Neural Networks to 3D-NoC-Based Neuromorphic Systems for Enhanced Reliability," *IEEE Access*, vol. 11, pp. 94664–94678, 2023. Conference Name: IEEE Access.

[39] J. Xue, L. Xie, F. Chen, L. Wu, Q. Tian, Y. Zhou, R. Ying, and P. Liu, "EdgeMap: An Optimized Mapping Toolchain for Spiking Neural Network in Edge Computing," *Sensors*, vol. 23, p. 6548, Jan. 2023. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[40] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020. Conference Name: IEEE Access.

[41] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, pp. 308–313, Jan. 1965.

[42] T.P. Runarsson and X. Yao, "Search biases in constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, pp. 233–243, May 2005. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[43] Y. Vesikar, K. Deb, and J. Blank, "Reference Point Based NSGA-III for Preferred Solutions," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1587–1594, Nov. 2018.

[44] Q. Zhang, S. Member, and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition."

[45] A. Benítez-Hidalgo, A. J. Nebro, J. García-Nieto, I. Oregi, and J. Del Ser, "jMetalPy: A Python framework for multi-objective optimization with metaheuristics," *Swarm and Evolutionary Computation*, vol. 51, p. 100598, Dec. 2019.

[46] F. Biscani and D. Izzo, "A parallel global multiobjective framework for optimization: pagmo," *Journal of Open Source Software*, vol. 5, p. 2338, Sept. 2020.

[47] J. Mejía, "Metaheuristics," Jan. 2023. original-date: 2017-10-29T05:05:37Z.

[48] C. Gagné and M. Parizeau, "Open BEAGLE: a new C++ Evolutionary Computation framework," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, (San Francisco, CA, USA), p. 888, Morgan Kaufmann Publishers Inc., July 2002.

[49] M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich, "Opt4J: a modular framework for meta-heuristic optimization," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, (Dublin, Ireland), p. 1723, ACM Press, 2011.

[50] T. Fühner and C. Jacob, *EvolVision — An Evolvica Visualization Tool*. Jan. 2001.

[51] V. N. Prithvish, "Efficient mapping of large scale SNN and rate-based DNN on SENeCA | TU Delft Repository."

[52] "Towards Efficient Deployment of Hybrid SNNs on Neuromorphic and Edge AI Hardware."

[53] S. H. Ahmed, J. Finkbeiner, and E. Neftci, "A Hybrid SNN-ANN Network for Event-based Object Detection with Spatial and Temporal Attention," Mar. 2024. arXiv:2403.10173 [cs].

[54] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948 vol.4, Nov. 1995.

[55] M. Rocklin, "Dask: Parallel Computation with Blocked algorithms and Task Scheduling," (Austin, Texas), pp. 126–132, 2015.

[56] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," Apr. 2016. Number: arXiv:1604.07316 arXiv:1604.07316 [cs].

[57] Y. Ho and D. Pepyne, "Simple Explanation of the No-Free-Lunch Theorem and Its Implications," *Journal of Optimization Theory and Applications*, vol. 115, pp. 549–570, Dec. 2002.

[58] S. Zhang, Z. Li, H. Zhou, R. Li, S. Wang, K.-W. Paik, and P. He, "Challenges and recent prospectives of 3D heterogeneous integration," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 2, p. 100052, Jan. 2022.