Communication Papers of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 41–47
DOI: 10.15439/2025F1427
ISSN 2300-5963 ACSIS, Vol. 45

Domain-as-Particle with PSO Methods for Neural-Network Feature Weighting

Fabio Berberi
University of Siena
Via Roma, 56, 53100 Siena, Italy
Leuphana University of Lueneburg
Universitätsallee 1, 21335 Lueneburg, Germany
ORCID: 0009-0004-8825-8707
Email: f.berberi@student.unisi.it

Paolo Mercorelli
Leuphana University of Lueneburg
Universitätsallee 1, 21335 Lueneburg, Germany
ORCID: 0000-0003-3288-5280
Email: paolo.mercorelli@leuphana.de

***Abstract*—We present a framework that integrates Particle Swarm Optimization (PSO), machine learning, K-Fold cross-validation, and surrogate modeling to identify optimal weight vectors for feature scaling in neural network training. In our approach, the n-dimensional weight space is partitioned into non-overlapping subdomains, each corresponding to a PSO particle. Particle movement is guided by a characteristic vector, which is determined by the best-performing candidates in each subdomain and by information exchanged with neighboring regions. To reduce evaluation costs, a surrogate model—trained on a uniformly sampled subset of candidates—pre-filters particles before full K-Fold validation. The top candidates then undergo comprehensive validation, updating the characteristic vectors for subsequent iterations. This domain-as-particle PSO framework enables efficient weight discovery, significantly reducing computational overhead while maintaining robust performance. The effectiveness of this approach is demonstrated on real-world datasets.**

***Index Terms*—Particle Swarm Optimization, surrogate modeling, neural networks, K-Fold cross-validation, feature weighting, medical application.**

I. Introduction

IDENTIFICATION methods play a pivotal role in the control and monitoring of dynamic systems. They form the backbone of modern engineering, seamlessly integrating concepts from various disciplines to achieve regulation, stabilization, and optimization of these systems. This multidisciplinary nature makes system identification and control a crucial area of both academic study and practical application, extending its impact across diverse fields such as engineering, computer science, mathematics, physics, biology, economics, and environmental engineering. Notable examples include contributions like [1] and [2], which demonstrate the successful application of identification and control techniques in environmental engineering and ecology. Similarly, [3] provides a practical case of identification methods applied within an economic and commercial context, specifically to prey-predator models.

Particle Swarm Optimization (PSO), first introduced in [4], is an evolutionary computation technique inspired by the social behaviors of bird flocking and fish schooling. Initially designed for continuous optimization problems within computational intelligence, PSO has evolved into a versatile, interdisciplinary tool applied across numerous domains. In computer science, it is utilized for tasks such as feature selection in machine learning [5] and hyperparameter optimization in deep learning models. The biomedical sector has also adopted PSO for applications like image segmentation [6] and gene expression data analysis, illustrating its adaptability to high-dimensional and noisy data environments. PSO's popularity can be attributed to its simplicity, flexibility, and effectiveness in addressing nonlinear, non-differentiable, and multi-objective problems, making it a powerful tool for both theoretical research and real-world applications. A detailed and recent survey of PSO can be found in [7]. Optimizing feature weights enhances neural network training by scaling inputs to accelerate convergence and improve accuracy. Manual feature engineering or exhaustive hyperparameter search becomes infeasible in high-dimensional spaces. We propose a *domain-as-particle PSO framework*, where the n-dimensional weight space, defined as the *mother domain* $\mathcal{W}$, is partitioned into distinct regions. Each region acts as a particle that independently explores its section to search for optimal solutions.

Contribution and Structure of the Paper

This paper presents a new framework for neural-network feature weighting based on a domain-as-particle Particle Swarm Optimization (PSO) strategy. The method efficiently partitions and explores the weight space, combining surrogate modeling with adaptive domain updates to enable scalable optimization in high-dimensional scenarios. The effectiveness of this approach is demonstrated through experiments on real-world datasets and benchmark comparisons.

The remainder of the paper is organized as follows. Section II presents the proposed method in detail. Section III describes the dataset and initialization procedures. Section IV discusses the experimental results. Section V concludes the paper and outlines future research directions.

II. Proposed Method

The proposed domain-as-particle Particle Swarm Optimization (PSO) framework for neural-network feature weighting

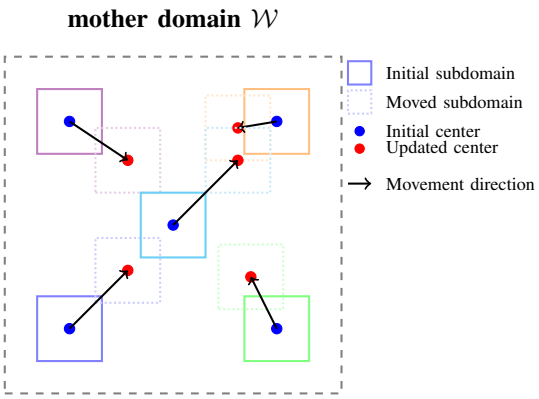

Fig. 1. Conceptual illustration of domain movement: five subdomains, initially uniformly distributed in $\mathcal{W}$, each move a short distance to new positions. The movement is driven by each subdomain's characteristic vector and neighbor influences.

operates through the following main steps:

- **partitioning:** The n-dimensional feature weight space is divided into multiple subdomains, each acting as a separate PSO particle.
- **Particle generation and surrogate modeling:** In each subdomain, a set of candidate weight vectors (particles) is generated. A subset is evaluated using neural network training and K-Fold cross-validation, and the results are used to train a surrogate model (ensemble of RandomForest and GradientBoosting regressors) that predicts the quality of remaining candidates.
- **Selection and validation of top candidates:** The surrogate model identifies the most promising particles, which are then fully validated using K-Fold cross-validation.
- **Characteristic vector update:** The best candidates in each subdomain are combined into a characteristic vector (e.g., weighted average) to guide the next search step.
- **Domain update (PSO movement):** Each subdomain's center is shifted according to its characteristic vector and, optionally, information from neighboring subdomains.
- **Iteration and stopping criteria:** These steps repeat iteratively, with surrogate retraining and domain adaptation, until convergence or budget limits are met.

This approach allows efficient and scalable search for optimal feature weightings in high-dimensional problems.

A. Weight-Space Partitioning

The weight-space partitioning is the foundational step of the proposed algorithm (see Fig. 2). The global weight space $\mathcal{W}$ is defined as the n-dimensional hypercube $[\mathrm{lb}, \mathrm{ub}]^n$, where n is the number of features and lb, ub are the lower and upper bounds for each weight, respectively. This weight space $\mathcal{W}$ is partitioned into several uniformly distributed subdomains D_j, each with fixed side length L and centered at a grid point $\mathbf{d}_j$. Importantly, these subdomains D_j do not overlap and have deliberate spacing between them.

This spacing allows each subdomain D_j room to move and adapt during optimization, facilitating thorough and independent exploration of the search space. As the algorithm iterates, each D_j shifts to investigate unexplored regions, ensuring efficient coverage of the entire main domain.

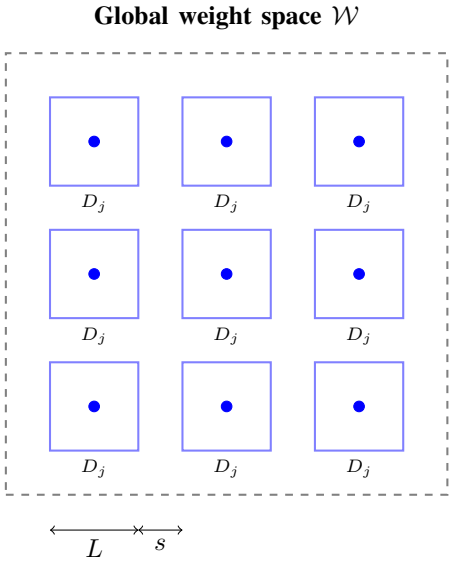

Fig. 2. Weight-space partitioning: $\mathcal{W}$ is partitioned into a 3×3 grid of subdomains D_j, each an n-dimensional cube of side length L, separated by spacing s, and offset by an internal margin M.

B. Particle Generation and Surrogate Modeling

After defining the subdomains (see Fig. 3), the next step is to generate candidate weight vectors inside each subdomain. For every D_j, both standard particles and those selected for surrogate modeling are **uniformly sampled** within the subdomain boundaries. This allows the algorithm to explore different weightings in each region of the search space.

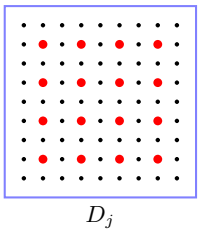

Fig. 3. Illustration of a single subdomain D_j with a uniform interior grid of black candidate particles (excluding the topmost row) and a uniformly spaced subset highlighted in red.

After generating both types of particles—the P full particles and the S surrogate particles—we begin the surrogate model construction as follows. For each surrogate particle, we estimate performance using K-Fold cross-validation: the dataset is split into K equal parts, with each part used once as the validation set and the others as the training set. The final performance metric is obtained by averaging results across all folds. This approach provides a robust estimate of the model's generalization ability.

With reference to the neural network (see Fig. 4): For each surrogate particle, the neural network is trained on the training folds and tested on the test fold, resulting in a composite value for each split. The composite value is defined as the arithmetic mean of three key classification metrics: accuracy, precision, and recall. This metric plays a central role in our algorithm as the primary objective to optimize, representing a holistic measure of the model's predictive power on the dataset.

Specifically, the composite value quantifies the overall quality of the candidate feature weighting by balancing multiple aspects of classification performance:

- **Accuracy** measures the proportion of correctly classified instances among all samples, providing a general assessment of model correctness.
- **Precision** evaluates the proportion of true positive predictions among all positive predictions made by the model, reflecting the model's ability to avoid false positives.
- **Recall** (also known as sensitivity) assesses the proportion of true positive cases correctly identified by the model, indicating its effectiveness at detecting positive instances.

Each of these metrics addresses a distinct aspect of classification performance, and combining them equally in the composite value ensures a balanced evaluation that mitigates biases inherent in any single metric. Maximizing the composite value during optimization directly corresponds to improving the model's ability to generalize and make accurate predictions, which is the ultimate goal of our feature weighting approach.

The hyperparameters and structural characteristics of the neural network used are not fixed but depend on factors such as the dataset size, the number of features, and other intrinsic properties of the data. Consequently, users applying this framework should perform preliminary experiments to identify a network configuration that produces high composite values from the earliest iterations, enhancing optimization efficiency.

In future work, we intend to augment this approach by leveraging Particle Swarm Optimization to automatically tune neural network hyperparameters, thus optimizing both feature weights and model architecture in a unified framework.

This procedure is repeated for all splits. At the end, the average performance is calculated for the model trained with the weight vector defined by the surrogate particle. This procedure is called full validation. This entire process is performed for every surrogate particle, so that for each surrogate particle, a composite value is generated. In other words, for each surrogate particle k in domain D_j, the K-Fold cross-validation returns a single composite value $\sigma_{j,k}$, representing the average performance of the neural network when trained and validated using the weight vector associated with that surrogate particle. At this stage, for each domain, the surrogate model is constructed by leveraging the composite values already computed for the surrogate particles. As illustrated in Fig. 5, the surrogate consists of a `RandomForestRegressor` and a `GradientBoostingRegressor`, both trained on (particle, composite value) pairs from K-Fold cross-validation.

```
{
  "MODEL_ARCHITECTURE": {
    "layer1_units": 124,
    "layer1_activation": "relu",
    "dropout1": 0.3,
    "layer2_units": 64,
    "layer2_activation": "relu",
    "dropout2": 0.3,
    "output_units": 1,
    "output_activation": "sigmoid"
  },
  "TRAINING": {
    "EPOCHS": 2,
    "batch_size": 16
  }
}
```

Fig. 4. Neural network configuration: two-layer feed-forward network with ReLU activation and dropout in the hidden layers, and a sigmoid activation in the output layer. The training is performed for 2 epochs with a batch size of 16. Both the architecture and all training hyperparameters are fully customizable and can be modified as needed for different applications.

The `RandomForestRegressor` operates by training a large number of independent decision trees, each built on different random subsets of the data and features. The final prediction is made by averaging the outputs of all the trees, which helps reduce overfitting and increases prediction stability.

The `GradientBoostingRegressor`, on the other hand, builds its decision trees sequentially. Each new tree is trained to correct the errors (residuals) of the combined predictions of the previous trees, gradually improving the overall predictive accuracy. This makes Gradient Boosting particularly effective at capturing complex patterns in the data.

For each candidate, the surrogate prediction is obtained by taking the mean of the outputs from the two regressors. By combining these two different regression techniques, the surrogate model provides fast, reliable, and robust estimates of candidate performance. This enables the algorithm to quickly identify and select the most promising candidates for expensive full validation with the neural network, significantly reducing computational costs.

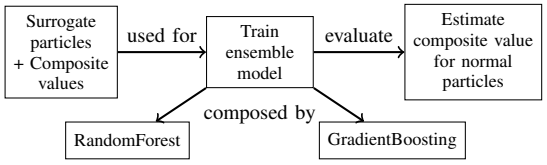

Fig. 5. Schematic of surrogate model creation: composite values from surrogate particles are used to train an ensemble consisting of RandomForest and GradientBoosting models. The predictions from both models are averaged to estimate the composite value for normal candidate particles.

C. Selection and Validation of Top Candidates

After the surrogate model predicts composite values for all candidate particles, a two-step selection process is applied,

as illustrated in Fig. 6. First, the top K particles with the highest surrogate-predicted scores are selected from the initial candidate set. Next, these selected candidates undergo full validation using K-Fold cross-validation with the actual neural network, in order to accurately assess their true performance.

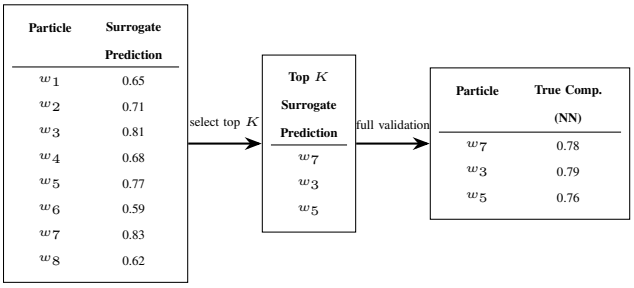

Fig. 6. Workflow of candidate selection: All candidate particles with surrogate predictions (left); selection of top K candidates (center); true composite values after full validation with the neural network (right).

D. Characteristic Vector Update

Once the top-K candidates $\{\mathbf{w}_{j,k}\}_{k=1}^{K}$ in each subdomain D_j have been fully validated, we aggregate their information into a single "characteristic" vector $\mathbf{c}_j$ that will guide the next movement of D_j. Let $\sigma_{j,k}$ denote the composite value assigned to candidate $\mathbf{w}_{j,k}$. We define:

$$\mathbf{c}_j = \frac{\sum_{k=1}^{K} \sigma_{j,k}\, \mathbf{w}_{j,k}}{\sum_{k=1}^{K} \sigma_{j,k}} \tag{1}$$

In other words, $\mathbf{c}_j$ is the weighted average of the K best weight vectors in D_j, with weights proportional to their observed performance. This choice has two desirable properties:

- *Exploit high-quality solutions:* vectors with larger $\sigma_{j,k}$ contribute more strongly, pulling $\mathbf{c}_j$ toward regions of higher performance.
- *Smooth update:* by averaging, we avoid abrupt jumps in domain center due to outlier candidates.

The updated characteristic vector $\mathbf{c}_j$ then serves as the *new position* (center) of subdomain D_j in the subsequent PSO iteration.

E. Domain Update (PSO Movement)

After defining the characteristic vector $\mathbf{c}_j$, each subdomain D_j is shifted by combining four components—*inertia*, *personal best*, *global best*, and, most critically, the *neighbor influence* that pulls D_j toward positions discovered by its adjacent domains. It is precisely this social component that enables our algorithm to operate in a PSO-like manner.

First, we introduce the three acceleration components:

$$\begin{aligned} \Delta_j^p &= \phi_p\, r_p \big(\mathbf{p}_j - \mathbf{c}_j^{(t-1)}\big), \\ \Delta_j^n &= \phi_n\, r_n \big(\mathbf{n}_j - \mathbf{c}_j^{(t-1)}\big), \\ \Delta_j^g &= \phi_g\, r_g \big(\mathbf{g} - \mathbf{c}_j^{(t-1)}\big). \end{aligned} \tag{2}$$

where:

- $\mathbf{p}_j$ is the personal best position of D_j,
- $\mathbf{n}_j$ is the best position discovered by its neighbors,
- $\mathbf{g}$ is the global best across all subdomains,
- $r_p, r_n, r_g \sim U(0,1)^n$ are independent uniform random vectors.

Next, the velocity is updated in PSO fashion:

$$\mathbf{v}_j^{(t)} = \underbrace{\omega\, \mathbf{v}_j^{(t-1)}}_{\text{inertia}} + \Delta_j^p + \Delta_j^n + \Delta_j^g, \tag{3}$$

and the subdomain center shifts according to

$$\mathbf{c}_j^{(t)} = \mathbf{c}_j^{(t-1)} + \mathbf{v}_j^{(t)}. \tag{4}$$

Fig. 7 illustrates how the characteristic vector (black arrow) and neighbor influences (red arrows) combine to drive each subdomain from its initial (blue solid) to its moved (blue dotted) position.

Interpretation of terms:

- $\omega\, \mathbf{v}_j^{(t-1)}$ (*inertia*): retains part of the previous velocity, smoothing motion.
- Δ_j^p (*cognitive/personal*): pulls the center toward $\mathbf{p}_j$, exploiting its own best.
- Δ_j^n (*social/neighbor*): pulls toward the neighbor best $\mathbf{n}_j$, enabling coordinated exploration.
- Δ_j^g (*global*): draws all subdomains toward the global best $\mathbf{g}$ for convergence.
- $r_p, r_n, r_g \sim U(0,1)^n$ (*random vectors*): introduce stochastic variation—larger values favor exploration, smaller values fine-tune exploitation.
- The update $\mathbf{c}_j^{(t)} = \mathbf{c}_j^{(t-1)} + \mathbf{v}_j^{(t)}$ completes the PSO move for D_j.

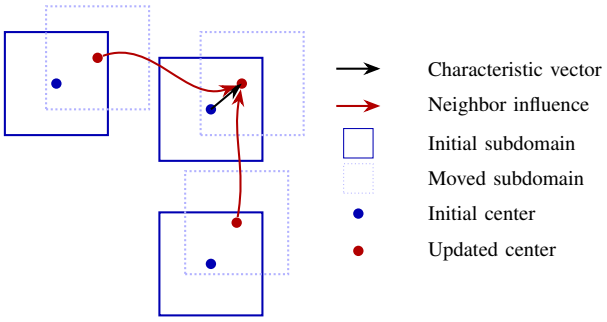

Fig. 7. Characteristic vector with neighbor influence: solid-line subdomains are initial, dotted are moved; centers shift from blue to red dots; black arrow shows characteristic vector; red arrows show neighbor influence.

Having detailed how each domain's center is updated via its characteristic vector, we now describe how these updates are organized over time and when the algorithm terminates.

F. Iteration and Stopping Criteria

Our framework proceeds in discrete iterations $t = 1, 2, \ldots$, mirroring the structure of classical PSO.

Per-iteration workflow:

1) *Domain movement*: each D_j is shifted by its updated velocity and characteristic vector.
2) *Surrogate set refresh*: the surrogate particles farthest from the new center are discarded and replaced by new samples drawn in the region just explored.
3) *Surrogate retraining*: the surrogate model for D_j is retrained on the updated surrogate set and their composite values.

Stopping criteria:

- *Target composite value*: terminate as soon as any candidate reaches a predefined composite score.
- *Fixed budget*: run for a preset number of iterations or wall-time, then return the best vector seen across all domains and iterations.
- *Budget exhaustion*: stop when the allotted computational resources (e.g. number of network evaluations or CPU-hours) are consumed.

Upon termination, the algorithm returns the highest-scoring weight vector found.

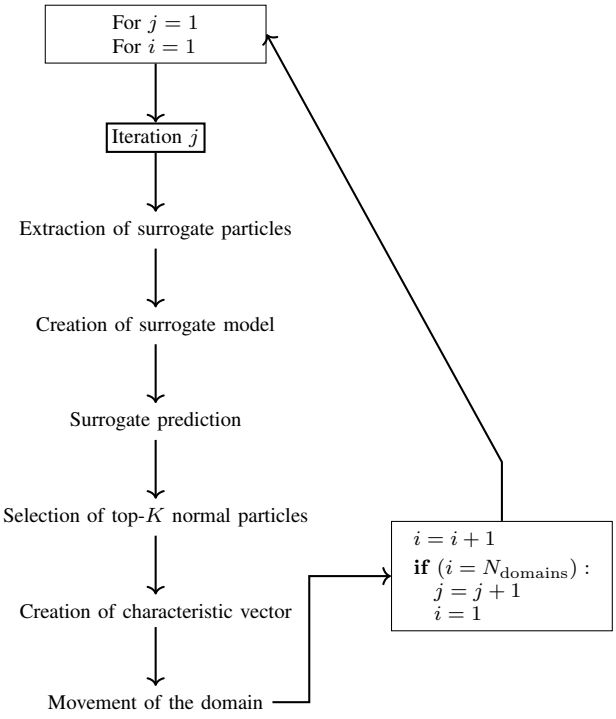

Fig. 8. Illustrative flow of the per-domain, per-iteration update loop: each iteration processes surrogate particle extraction, surrogate model training, prediction, top-K selection, characteristic-vector creation, and domain movement, followed by index updates. Here N_{domains} denotes the total number of subdomains within the parent domain.

III. Dataset and Initialization Procedures

The proposed framework was validated on the Wisconsin Diagnostic Breast Cancer (WDBC) dataset, which includes $n = 30$ cellular features per sample and binary labels indicating malignant (M) or benign (B) tumors. The dataset consists of $N = 569$ samples; the label column was converted to 1 (malignant) and 0 (benign) for classification.

All features were standardized using z-score normalization. The dataset was split into training and test sets with an 80-20 stratified split to preserve the original class balance. Categorical labels were mapped as follows: $M \to 1$, $B \to 0$.

The main phases of data preparation and the per-domain optimization cycle are illustrated in Fig. 8.

For class imbalance, balanced class weights were computed and applied during training.

The PSO-based optimization was initialized as follows:

- Global weight bounds: $\text{lb} = 1.0$, $\text{ub} = 10.0$
- Number of subdomains: $m = 10$
- Subdomain width: $L = 0.5$
- Particles per domain: $P = 100$
- Surrogate particles per domain: $S = 10$
- Precision: weights rounded to one decimal
- Domain update factor: $\gamma = 0.1$
- Number of global iterations: $T = 5$
- Early stopping threshold: $\epsilon = 0.001$
- Top $K = 20$ particles per domain revalidated

Within each subdomain, candidate weight vectors (particles) were generated using Latin Hypercube Sampling and sorted by the sum of their components. Surrogate models (Random Forest and Gradient Boosting Regressors) were trained on a uniform subset of particles, allowing efficient pre-selection before full neural network validation.

The neural network used was a feed-forward model with two hidden layers (124 and 64 units, ReLU activation, 0.2 dropout), trained using the Adam optimizer and binary cross-entropy loss, following established approaches that apply sensitivity and feature-importance analysis for medical diagnosis using neural networks [8]. Performance was assessed via 5-fold cross-validation, averaging accuracy, precision, and recall into a composite metric.

IV. Experimental Results

To rigorously assess the performance of our domain-as-particle PSO framework, we conducted a direct comparison with the standard (classical) PSO approach on the widely studied Wisconsin Diagnostic Breast Cancer (WDBC) dataset. Both methods were evaluated under challenging, high-dimensional conditions ($n = 30$), which are typical of real-world feature weighting problems.

Figure 9 displays the composite value trajectories for each subdomain during the optimization process using our proposed framework. In this experiment, we initialized 1000 particles per domain (i.e., per subdomain), adopted a batch size of 30 samples per neural network update, and used the neural network architecture described in Figure 4. The plot shows the evolution of composite values for all 10 domains distributed throughout the search space. Notably, subdomain 6 (highlighted in orange) achieved the highest composite value, indicating that this region of the weight space contained weight vectors that assigned the most effective feature scaling to the dataset. In contrast, the other domains, which do not

reach such high composite values, are evidently located in less favorable regions of the 30-dimensional search space, where the initialized weights do not lead to optimal model performance.

For the baseline comparison, Figure 10 presents results obtained using the classical PSO algorithm, without any subdomain partitioning or surrogate modeling. In this setting, we distributed 100,000 particles across the entire weight space to achieve comprehensive coverage of the 30-dimensional space, and used a batch size of 40 samples. Importantly, this approach does not leverage any of the domain-as-particle or surrogate-assisted strategies introduced in our framework. As a consequence, the algorithm must directly evaluate every candidate weight vector with full neural network training, leading to a substantial increase in computational burden and runtime.

From a computational perspective, the efficiency gain provided by our method is substantial. While the classical PSO required approximately hours to complete, our domain-as-particle approach achieved comparable or superior composite values in just 3 hours. This demonstrates not only a significant reduction in runtime, but also a clear advantage in terms of computational cost and scalability. The improvement in both efficiency and solution quality highlights the effectiveness of the proposed surrogate-assisted, multi-domain optimization strategy for neural-network feature weighting. To further benchmark our method, we implemented a simple attention-based classifier. The model consists of a self-attention layer with dimensionality equal to the number of input features (30), followed by two fully connected layers with ReLU activations and a sigmoid output layer for binary classification. The classifier was trained with the Adam optimizer at a learning rate of 1×10^{-3}, a batch size of 32, and for 50 epochs. This attention-based model achieved a composite value of approximately 0.94, which is slightly lower than the composite value obtained by our proposed domain-as-particle PSO framework, demonstrating that our method outperforms this baseline in terms of feature weighting effectiveness.

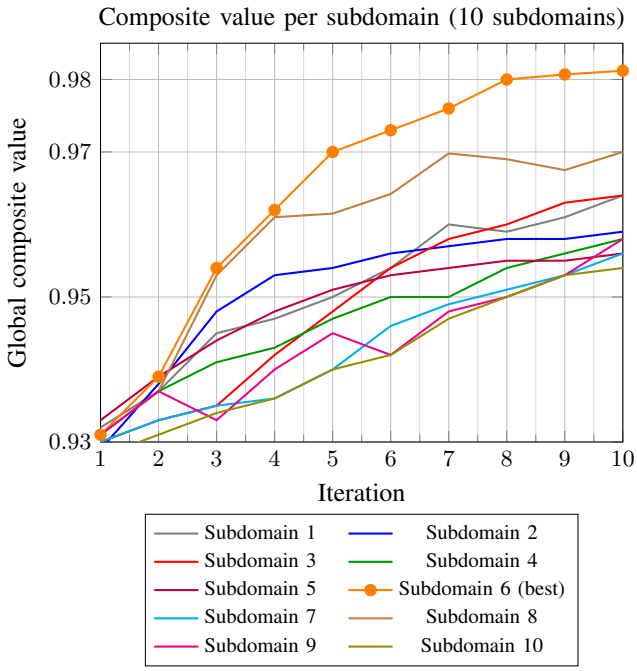

Fig. 9. Composite value per subdomain obtained using the domain-as-particle PSO method (multi-domain, surrogate-assisted).

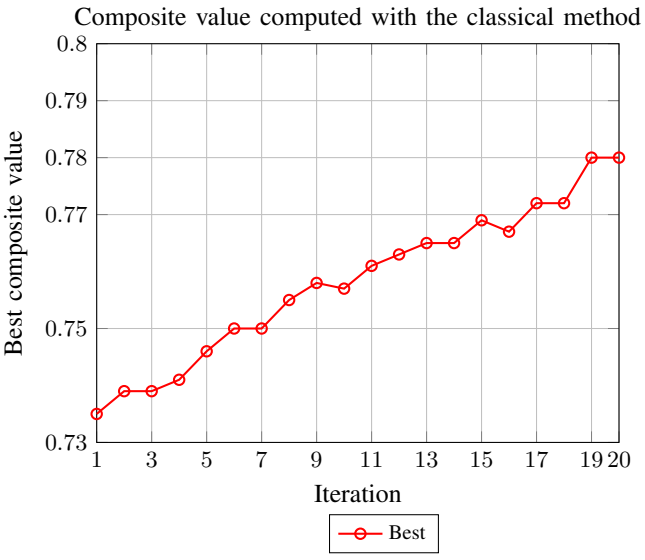

Fig. 10. Best composite value per iteration for the classical PSO approach (single-domain, no surrogate model).

V. Conclusion and Future Work

This paper introduced a domain-as-particle Particle Swarm Optimization (PSO) framework for neural network feature weighting, validated on medical data for breast cancer diagnosis. The method leverages surrogate models to efficiently guide the optimization process, achieving fast convergence to high-quality solutions with reduced computational cost. Our framework demonstrated robustness and scalability, making it well-suited for complex, high-dimensional feature weighting problems.

The algorithm has been tested on multiple datasets beyond the one presented here, consistently producing promising results that validate its general applicability and effectiveness. We plan to extend this line of research by applying the method

to a wider variety of datasets from diverse domains, to further confirm its versatility and real-world impact.

As a significant avenue for future work, we propose an adaptive extension of the algorithm that incorporates automatic hyperparameter tuning using PSO itself. Specifically, this variant will treat the hyperparameters of the Domain-as-Particle algorithm as an n-dimensional search space, where n is the number of hyperparameters to optimize. PSO will then seek the vector of hyperparameter values that yields the best composite value from the very first iteration, effectively initializing the algorithm in a data-driven, intelligent manner. This approach will allow the algorithm to automatically adapt to the characteristics of each dataset, significantly reducing the time and effort required to manually tune hyperparameters and accelerating convergence toward optimal solutions.

Moreover, we plan to investigate the introduction of a *grandmother domain* concept, a hierarchical multi-level structure within which multiple mother domains operate. This layered framework will enable optimization across larger and more complex weight spaces by coordinating exploration at different granularities, potentially enhancing solution quality and search efficiency.

Finally, future developments will explore new surrogate modeling techniques and innovative exploration strategies to further improve performance and scalability. These advancements aim to broaden the applicability of our framework and address emerging challenges in neural network optimization for real-world applications.

ACKNOWLEDGMENT

This work was inspired by the lecture held by Prof. Paolo Mercorelli entitled: “Applied Algorithms in Estimation and in Control of Technical, Economical, and Biological Dynamical Systems” within the scope of the Complementary Studies Programme at Leuphana University of Lueneburg during the winter semester 2024–2025. In this framework, students can explore other disciplinary and methodological approaches from the second semester onwards, focusing on additional aspects in parallel with their subjects and giving them the opportunity to sharpen skills across disciplines.

REFERENCES

[1] K. Benz, C. Rech, and P. Mercorelli, “Sustainable management of marine fish stocks by means of sliding mode control,” in *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems*, ser. FedCSIS 2019, vol. 18. IEEE, Sep. 2019. doi: 10.15439/2019f221. ISSN 2300-5963 p. 907–910. [Online]. Available: http://dx.doi.org/10.15439/2019F221

[2] K. Benz, C. Rech, P. Mercorelli, and O. Sergiyenko, “Two cascaded and extended Kalman filters combined with sliding mode control for sustainable management of marine fish stocks,” *Journal of Automation, Mobile Robotics and Intelligent Systems*, p. 28–35, Jul. 2019. doi: 10.14313/jamris/3-2020/30. [Online]. Available: http://dx.doi.org/10.14313/JAMRIS/3-2020/30

[3] D. Normatov and P. Mercorelli, “Parameters estimation of a Lotka-Volterra model in an application for market graphics processing units,” in *2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2022. doi: 10.15439/2022F61 pp. 935–938.

[4] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, vol. 4, 1995. doi: 10.1109/ICNN.1995.488968 pp. 1942–1948 vol.4.

[5] B. Xue, M. Zhang, W. N. Browne, and X. Yao, “A survey on evolutionary computation approaches to feature selection,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016. doi: 10.1109/TEVC.2015.2504420

[6] Y. Zhang, L. Wu, and S. Wang, “Magnetic resonance brain image classification by an improved artificial Bee Colony Algorithm,” *Progress In Electromagnetics Research*, vol. 116, p. 65–79, 2011. doi: 10.2528/pier11031709. [Online]. Available: http://dx.doi.org/10.2528/PIER11031709

[7] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, “Particle swarm optimization: A comprehensive survey,” *IEEE Access*, vol. 10, pp. 10 031–10 061, 2022. doi: 10.1109/ACCESS.2022.3142859

[8] P. A. Kowalski and M. Kusy, “Determining the significance of features with the use of sobol’ method in probabilistic neural network classification tasks,” in *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2017. doi: 10.15439/2017F225 pp. 39–48. [Online]. Available: https://annals-csis.org/Volume_11/drp/225.html