Automatic Parameter Selection for the PSO Algorithm Using Recurrent Neural Networks

Eduardo Santizo, Luis Alberto Rivera
Department of Electronics, Mechatronics and Biomedical Engineering
Universidad del Valle de Guatemala
Guatemala, Guatemala
{san16089, larivera}@uvg.edu.gt

***Abstract*—The Particle Swarm Optimization (PSO) algorithm is an stochastic optimization method that deploys a swarm of particles to explore and find the minimum of a cost function. In its most basic form, the algorithm can diverge depending on the parameters used. Two common solutions for this instability problem is the addition of an inertial constant and the constriction of the parameters through a group of equations that guarantee the convergence of the algorithm. The problem with these approaches is that they themselves are dependent on a group of parameters that need to be chosen carefully. In order to obtain a truly "off-the-shelf" solution, we propose the use of the PSO Tuner, a specially trained recurrent neural network (RNN) that automatically generates the value for these parameters. Three different architectures were tested for the network, each one based on a different type of neuron: GRU, LSTM and BiLSTM. After training, testing and comparing each architecture with each other, as well as with the standard PSO, the BiLSTM based PSO Tuner returned the best results in terms of speed and precision of convergence, as well as computation time, proving to be a viable solution for the automatic parameter selection problem.**

***Index Terms*—PSO, Recurrent Neural Networks, PSO Tuner**

I. Introduction

The PSO algorithm is an stochastic optimization method based on the movement seen in flocks of birds. In it, a group of particles, each with their own position and speed, explore a cost function to find the coordinates of its minimum. In its base form, the algorithm can diverge depending on the selection of its parameters. To solve this, two solutions are commonly proposed: The introduction of an inertial constant ω [1], and the constriction of its cognitive and social components through the use of equations dependent on constants ϕ_1 and ϕ_2 [2].

Both solutions prevent divergence by restricting the speed values for the particles. However, with the introduction of these parameters, comes the need for a new set of criteria to select their values. For ϕ_1 and ϕ_2, [2] notes that convergence is guaranteed while $\phi_1 + \phi_2 \geq 4$. In the case of ω, there are countless methods to determine its value, like those presented in [3], [4], [5] and [6].

In other words, there is not a surefire method to select the value of these parameters, only guidelines that the user has to take into account. This is why the main goal of this work is to design a method to automatically select the value of ϕ_1, ϕ_2 and ω.

This was achieved through the use of the PSO Tuner, a recurrent neural network (RNN) that takes in a group of metrics that describe the characteristics of the PSO swarm and outputs an estimate of the parameters ϕ_1, ϕ_2 and ω for each time step. Three types of RNN neurons were tested for the network's architecture: GRU, LSTM and BiLSTM. Each variation was trained with a total of 7700 runs of the standard PSO algorithm using different types of restrictions (inertia, constriction and a mixture of both) and cost functions.

To validate the effectiveness of this method, we measured the rate and precision of convergence of the PSO algorithm, as well as the computation time of the PSO Tuner with each type of neuron. The goal was to examine if the PSO Tuner brought improvements to the PSO algorithm and in case it did, determine which type of neuron generated the best and fastest results.

The paper is organized as follows: Background is presented in Section II. Section III shows the structure and implementation of the PSO Tuner. Section IV, shows the results of the validation experiments. A discussion on the *modus operandi* of the PSO Tuner is presented in Section V. Conclusions are presented in Section VI.

II. Background

A. Particle Swarm Optimization (PSO)

The *Particle Swarm Optimization* (PSO) algorithm presented by [1] proposes the creation of a set of "m" particles, each with a corresponding position and speed vector. These particles move on the surface of a target function whose independent variables consist of the "n" coordinates of each particle. This objective function is called a *cost function*, and the scalar resulting from evaluating the coordinates of each particle in the cost function is called *cost*.

The objective of the particles is to find the set of coordinates that generate the smallest possible cost value within a given search region. To do this, the particles are placed in random starting positions and then evaluate their coordinates in the cost function. If the cost is lower than in their previous position, the particle is said to have found a new *personal best* ($\overrightarrow{p_{pos}}(t)$).

This process is repeated for each particle in the swarm, resulting in "m" $\overrightarrow{p_{pos}}(t)$ coordinates. The $\overrightarrow{p_{pos}}(t)$ that corresponds to the lowest cost out of all "m" estimates is known as the current *global minimum*. If the current *global minimum*

cost is lower than that of the previous iteration, the swarm is said to have found a new *global best* ($\overrightarrow{g_{pos}}(t)$).

After determining their cost and best positions, all particles update their velocity vector (given an initial random velocity) and position making use of the following set of equations:

$$\begin{aligned}\overrightarrow{v}(t+1) &= \overrightarrow{v}(t) \\ &+ R_1C_1(\overrightarrow{p_{pos}}(t) - \overrightarrow{x}(t)) \quad \textit{Cognitive Comp.} \\ &+ R_2C_2(\overrightarrow{g_{pos}}(t) - \overrightarrow{x}(t)) \quad \textit{Social Comp.} \\ \overrightarrow{x}(t+1) &= \overrightarrow{x}(t) + \overrightarrow{v}(t+1)\end{aligned} \tag{1}$$

R_1, R_2 : Uniform random numbers between 0 & 1

C_1, C_2 : Acceleration coefficients

Because the *personal best* comes from each particle's individual memory of its best position so far, the term of the velocity equation that uses $\overrightarrow{p_{pos}}(t)$ is called the *cognitive component*. On the other hand, due to the *global best* coming from the collective memory of the best position found so far by the whole swarm, the term of the speed equation that uses $\overrightarrow{g_{pos}}(t)$ is called the *social component*.

After the particles move, the process repeats itself all over again. The algorithm continues to iterate until all particles converge to a single set of coordinates, which is said to be the minimum of the cost function.

B. Restrictions

The initial PSO algorithm lacked constraints for the velocity vector, leading to an oscillatory or divergent behavior in certain situations. To prevent this, [1] manually set a maximum value for the velocity components in order to restrain their magnitude between $(-V_{max},\ V_{max})$. This produced better results, but the selection of the parameter V_{max} required a great deal of testing in order to obtain an optimal value for the given application.

Shortly thereafter, [7] proposed the addition of a new coefficient ω to equation 1 as a way to prevent divergence.

$$\begin{aligned}\overrightarrow{v}(t+1) &= \omega\overrightarrow{v}(t) \quad \textit{Inertial Term} \\ &+ R_1C_1(\overrightarrow{p_{pos}}(t) - \overrightarrow{x}(t)) \quad \textit{Cognitive Comp.} \\ &+ R_2C_2(\overrightarrow{g_{pos}}(t) - \overrightarrow{x}(t)) \quad \textit{Social Comp.}\end{aligned} \tag{2}$$

This term was called the inertial coefficient, because it represents the apparent "fluidity" of the medium in which the particles move. The idea of this modification was to start the algorithm with a high fluidity (ω = 0.9), linearly decreasing ω until a lower bound is reached (ω = 0.4). This favored exploration at the start, while simultaneously encouraging clustering and convergence towards the end of the algorithm.

The introduction of the now called *linearly decreasing inertia* brought many improvements to the PSO algorithm and paved the way to many other types of inertia now commonly used among the scientific community [3]. However, the algorithm still suffered from stability problems in some edge cases.

This is why [2] modeled the algorithm as a dynamic system. After analyzing the stability of its model, it was concluded that, as long as its eigenvalues met certain relationships, the particle system would always converge. One of the most computationally efficient relationships to implement was the *Model Type 1'*, a modification to equation 1 based on the addition of the constriction coefficients ϕ_1 and ϕ_2.

$$\begin{aligned}\overrightarrow{v}(t+1) &= \chi\,(\overrightarrow{v}(t) \\ &+ \overrightarrow{U}(0,\phi_1)\,(\overrightarrow{p_{pos}}(t) - \overrightarrow{x}(t)) \\ &+ \overrightarrow{U}(0,\phi_2)\,(\overrightarrow{g_{pos}}(t) - \overrightarrow{x}(t)))\end{aligned} \tag{3}$$

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \qquad \phi = \phi_1 + \phi_2 \tag{4}$$

$\overrightarrow{U}(0,x)$: Uniform random numbers between 0 & x

As a general rule, $\phi \geq 4$ to ensure convergence. Usually, $\phi_1 = \phi_2 = 2.05$ and $\kappa = 1$. Its worth mentioning that in this new set of equations, χ takes the place of the inertial coefficient ω. This means that the reintroduction of the standard inertial coefficient would be detrimental to the algorithm, since it introduces unexpected effects on its stability.

However, [8] discovered that by mixing both restriction methods (inertial and constriction), improvements can be made to both the speed and accuracy of the PSO algorithm.

$$\begin{aligned}\overrightarrow{v}(t+1) &= \chi\,(\omega\overrightarrow{v}(t) \\ &+ \overrightarrow{U}(0,\phi_1)\,(\overrightarrow{p_{pos}}(t) - \overrightarrow{x}(t)) \\ &+ \overrightarrow{U}(0,\phi_2)\,(\overrightarrow{g_{pos}}(t) - \overrightarrow{x}(t)))\end{aligned} \tag{5}$$

Another inherent advantage of this "mixed" restriction method is that it enables the user to use any of the restriction methods depending on the value of ϕ_1, ϕ_2, χ and ω. For the inertial restriction method, $\chi = \phi_1 = \phi_2 = 1$. For the constriction method, $\omega = 1$. Finally for the mixed method used by [9], $\phi_1 = 2$, $\phi_2 = 10$ and ω is set as an *exponentially decreasing inertia type*.

III. PSO Tuner

The PSO variation that uses equations 4 and 5 is dubbed MPSO [8], and its dependent upon three parameters: ϕ_1, ϕ_2 and ω. ϕ_1 and ϕ_2 can be selected according to the inequality $\phi_1 + \phi_2 \geq 4$, while the type of inertia ω can be selected according to the application needs. However, the user would need to test a large number of parameter combinations in order to find the combination that results in an optimal behavior.

To automate this task, a neural network was devised. Named *PSO Tuner*, the neural network takes the current "*state*" of the PSO swarm, and then generates a time varying estimate for the parameters of the MPSO. Due to the time dependent

nature of the generation process, a recurrent neural network was selected, as it is capable of taking into account past information in order generate better present results.

A. Network Inputs

The neural network requires a quantitative measurement of the current swarm state as an input. This could be accomplished by feeding the network all the positions and velocities of the swarm, but this solution lacks scalability. If the dimensionality of the *input vector* becomes dependent on the number of particles, for every change in swarm size, a new network would need to be trained.

To avoid this problem, the swarm state was measured through a set of 4 metrics independent from swarm size: Normalized average of the average distance between all particles (D_{all}^{N}), coherence (S_c), average standard deviation ($\overline{\sigma}^{N}$) and normalized "goal to global best" distance (D_{gb}^{N}). All metrics were designed or normalized to produce values between 0 and 1. Concatenating all the metrics vertically, the resulting *input vector* took the form of a 4×1 array.

$$\text{Input Vector} = \begin{bmatrix} \overline{\sigma}^{N} \\ S_c \\ D_{gb}^{N} \\ D_{all}^{N} \end{bmatrix}$$

B. PSO Metrics

1) Average of the Average Distance Between all Particles: For every particle in the swarm, the average distance to all other particles is computed. All averages are then averaged to obtain a single scalar value per swarm (6). This metric measures the relative dispersion of all particles with respect to their neighbors. Due to the addition of a second average, the effect of outliers is minimized [10].

$$\mathcal{D}_{all} = \frac{1}{|S|}\sum_{i=1}^{|S|}\left(\frac{1}{|S|}\sum_{j=1}^{|S|}\sqrt{\sum_{k=1}^{|K|}\left(x_{ik}-x_{jk}\right)^2}\right) \tag{6}$$

$|S|$: Swarm size

$|K|$: Dimensionality of the cost function

To restrict the possible output values between 0 and 1, the scalar $\mathcal{D}_{all}$ was normalized using the maximum distance possible between the current global minimum of the cost function and the corners of the search region given to the PSO particles[1]. This distance (d_{max}) is also later used to normalize D_{gb}.

$$D_{all}^{N} = \frac{D_{all}}{d_{max}} \tag{7}$$

[1]For the two-dimensional case, the search region consists of a square. For higher dimensionality problems, the search region consists of a hypercube.

2) Coherence: This metric consists of the ratio between the speed of the swarm center $\mathbf{v}_s$ and the average speed of the particles $\overline{\mathbf{v}}$ (8). Coherence values range between 0 and 1.

$$S_c = \frac{\mathbf{v}_s}{\overline{\mathbf{v}}} \tag{8}$$

$$\mathbf{v}_s = \frac{1}{|S|}\left\|\sum_{i=1}^{|S|}\tilde{\mathbf{v}}_i\right\|_2 \tag{9}$$

$$\overline{\mathbf{v}} = \frac{1}{|S|}\sum_{i=1}^{|S|}\|\tilde{\mathbf{v}}_i\|_2 \tag{10}$$

$\|v\|_2$: Euclidean norm or 2-norm

A high coherence value can be generated by a high swarm center velocity or a low average velocity. Therefore, a high coherence is a sign of convergence (low average velocities) or of a coordinated movement of the particles in a specific direction (all individual velocities share directions).

A low value, can be generated by a high average velocity or a very low center velocity. This means that a low coherence is a sign of scattering (very high average velocities per particle) or stagnation of the swarm center (particle velocities cancel each other out) [10].

3) Average Standard Deviation: This is defined as the average standard deviation over all $|K|$ dimensions of the cost function (11). Its values range from 0 to 1. Used by [8] to quantify swarm dispersion.

$$\overline{\sigma} = \frac{1}{|K|}\sum_{k=1}^{|K|}\sqrt{\frac{\sum_{i=1}^{|S|}\left(x_{ik}-\overline{\mathbf{x}}_k\right)^2}{|S|}} \tag{11}$$

4) Normalized Distance of Goal to Global Best: This metric consists of the euclidean distance between the minimum of the cost function (x_{goal}, y_{goal}) and the swarm's *global best* (x_{gb}, y_{gb}). For its normalization, the distance is divided by d_{max} (12). For the two-dimensional case, this is calculated as follows

$$\begin{aligned} d_{gb} &= \|(x_m - x_{gb}), (y_m - y_{gb})\|_2 \\ D_{gb}^{N} &= \tfrac{d_{gb}}{d_{max}} \end{aligned} \tag{12}$$

C. Network Outputs

The output vector retains the same shape of the input vector: a 4×1 array.

$$\text{Output Vector} = \begin{bmatrix} \omega \\ \phi_1 \\ \phi_2 \\ \text{Iter}^{N} \end{bmatrix}$$

The first three rows consist of the MPSO parameters ω, ϕ_1 and ϕ_2, while the last row consists of a normalized estimate of the total number of iterations the PSO algorithm will take to converge.

$$\text{Iter}^N = \frac{\text{Iter}_{conv}}{\text{Iter}_{max}} \tag{13}$$

Iter_{conv} : Iterations used to reach convergence
Iter_{max} : Max. iteration number set by user

D. Training Data

For the network's training data, a large number of MPSO simulations were done while varying 3 initial conditions (as seen in Table I): the cost function minimized, the restriction method used and the type of inertia employed (in case the restriction method is set to inertia).

Initial Condition	Options
Cost Function	Banana. Dropwave. Levy N13. Himmelblau. Rastrigin. Schaffer F6. Sphere. Booth. Ackley. Styblinski-Tang. Griewank.
Restriction	Inertia. Constriction. Mixed.
Inertia Type	Constant. Linear. Chaotic. Random. Exponentially Decreasing.

TABLE I
OPTIONS USED FOR EVERY INITIAL CONDITION VARIED DURING THE TRAINING DATA GENERATION PROCESS

For every iteration in the simulation, a time step of the PSO algorithm was computed, the 4 input metrics were calculated and the value of the MPSO parameters (ω, ϕ_1 and ϕ_2) was noted. In case the particles converge at a specific place in the cost function (not necessarily the global minimum), the algorithm was terminated and the input and output vectors were constructed. This process was repeated a total of 100 times for every possible permutation of the initial conditions, leading to a total of 7700 data sequences being created for both the input and output vectors.

Said process was repeated a second time for the generation of the *validation data*, reducing the number of simulations per permutation of the initial conditions to 20. With this modification, 1540 pairs of input and output validation sequences were generated.

The training process was done using the *Deep Learning Toolbox* provided by Matlab.

E. Neuron Types

Matlab's *Deep Learning Toolbox* offers three types of recurrent neural network neurons for training: GRU, LSTM and BiLSTM. In terms of their computation capacity, BiLSTM neurons are the most "powerful", being able to learn patterns using both past and future information. These are followed by LSTM neurons, only capable of using past information to generate their estimates. Finally, there are the GRU neurons, which were specifically designed to be a simplification of the LSTM neuron, and therefore have a lower computation capacity.

It was decided to experiment with these three types of neurons to see if their learning ability varied according to the previously described expectations. In addition, since the internal complexity of each neuron decreases with each decrease in the capacity hierarchy (higher computational load for BiLSTM and lower for GRU), it was desired to observe which was the simplest neuron capable of generating satisfactory results.

F. Architecture and Training

The architecture of a neural network has a large number of hyperparameters with which the researcher can experiment: *number of layers*, *number of neurons per layer*, *activation functions*, *epochs*, *mini batch size*, etc. This is why in the case of the PSO Tuner, the process of adjusting these hyperparameters led to an extensive trial and error process in which the parameters were slightly altered, the network was trained and the effects of the change were then analyzed by performing a simulation of the MPSO algorithm.

Said simulations were conducted by directly connecting the output of the network to the parameters of the MPSO algorithm. After a few training tests, two interesting behaviors were observed. If the network was given control over ω, the swarm would start to display an erratic behavior in some cases. At the same time, if the user provided more than one input vector in each iteration (concatenating multiple input vectors in the form of a sequence), the algorithm seemed to dampen this unstable behavior.

This is why two additional systems were introduced as part of the PSO Tuner: The ability to disable the network's control over the parameter ω, and a way to feed more than one "sample" of the input vector to the network architecture. With both systems implemented, the training and hyperparameter tuning process resulted in the three network architectures presented in Table II. Their training parameters and use options (sample number and control over inertia) are detailed as well.

IV. Experiments and Results

To measure the effectiveness of the PSO Tuner (PSOT), three different aspects of the MPSO algorithm were tested: dispersion, rate and precision of convergence, and computation time. For every test, a total of 200 MPSO simulations were run, using a 1000 particles and changing the type of PSO Tuner used (GRU, LSTM, BiLSTM and None). Said runs were then averaged and displayed as figures. 9 total averages were obtained for each test, each one using a different combination of restriction type (inertia, constriction and mixed) and cost function (Griewank, Schaffer F6 and Sphere).

A. Swarm Dispersion

After the introduction of the PSO Tuner, swarm center movements tend to display one of three behaviors: irregular patterns, oscillations or a smooth fall towards a minimum (Figure 1). In either case, what the introduction of the PSO

Neuron Type	Architecture	Training Options							
		Optimizer	Max Epochs	Mini Batch Size	Initial Learn Rate	Learn Rate Drop Period	Learn Rate Drop Factor	No. of Samples	Inertia Enabled?
GRU	Input Layer (4) GRU Neurons (200) Dropout Layer (20%) Fully Connected (150) Fully Connected (100) Output Layer (4) Regression Layer	ADAM	100	100	0.001	2000	0.2	1	No
LSTM	Input Layer (4) LSTM Neurons (200) Dropout Layer (40%) Fully Connected (100) Fully Connected (50) Output Layer (4) Regression Layer	ADAM	100	100	0.001	2000	0.2	2	No
BiLSTM	Input Layer (4) BiLSTM Neurons (150) Fully Connected (50) Dropout Layer (50%) Output Layer (4) Regression Layer	ADAM	100	10	0.001	2000	0.2	1	Yes

TABLE II
ARCHITECTURE AND TRAINING OPTIONS FOR THE GRU, LSTM AND BILSTM NETWORKS USED FOR THE PSO TUNER

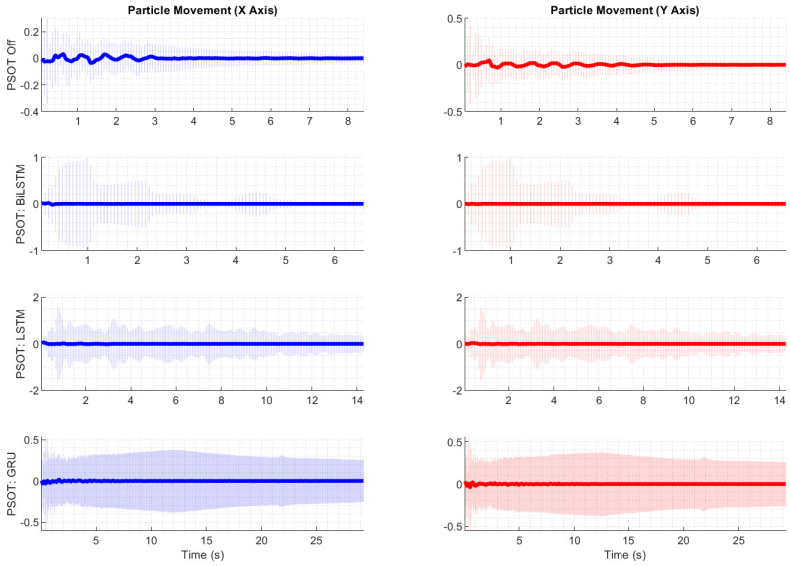

Fig. 1. Dispersion (transparent color) and median (solid line) of the PSO swarm using the "*Griewank*" cost function and "Inertia" restriction.

Tuner does is to smooth the movement of the center of the swarm. The quality of the smoothing is dependent on the architecture used.

If analyzed within the context of signal processing, the BiLSTM network seems to reduce the overshoot and noise of the movement, while increasing its rise time. This translates into a much more *purposeful* movement that is characterized by seeking the global minimum of the cost function rather than orbiting around it. This coincides with the knowledge of BiLSTM neurons, which indicates that they use both past and future information to generate their estimates. In simpler terms, the smoothing occurs because the network knows where it is going.

The smoothing of the remaining networks is slightly worse, retaining some fragments of the original noise or oscillations. In the case of the GRU network, the oscillations are translated into zigzag movements as a consequence of the erratic movement that accompanies this type of network. In the case of the LSTM network, it does not seem to filter the movement completely, but simply attenuates the sudden changes in the trajectory. In other words, the network knows where it must go, but it hesitates on the way.

The dispersion does not seem to exhibit a defined pattern, since its tendency varies according to the cost function and the

type of restriction. A constant in all cases is the lack of dispersion in the mixed restriction method. This is because, this restriction method produces a joint movement of the particles towards the first minimum found almost immediately. Since all PSO Tuner variations seem to imitate this behavior with particular intensity, it is to be expected that their movement presents an almost identical pattern.

B. Speed and Precision of Convergence

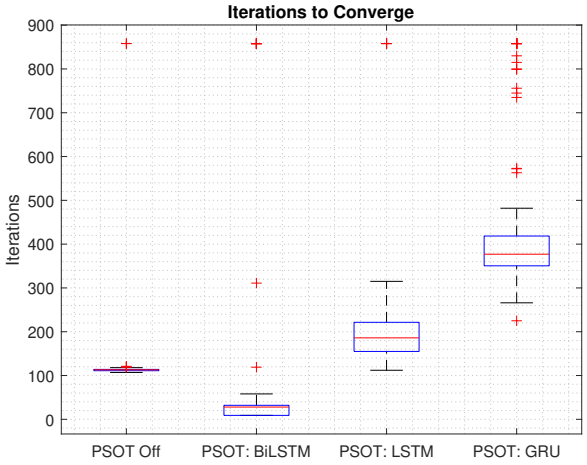

Fig. 2. Convergence speed (iterations to converge) of the PSO algorithm with and without the PSO Tuner (PSOT) enabled, using the "*Griewank*" cost function and "Inertia" restriction for the MPSO algorithm.

In terms of speed of convergence, the BiLSTM network is presented as the fastest alternative out of the three architectures tested (as seen in Figure 2). The only case where it is not presented as the superior option is in the cost function *Sphere*, in conjunction with the mixed restriction method. In terms of the slowest option, the GRU network was presented as the alternative with the longest convergence time in almost all cases. The exceptions to this rule consist of the cases where the constriction method is used, which seems to favor the correct operation of the GRU network. The LSTM network tends to have a convergence time equal to or greater than that of the default MPSO. In the few cases where the average number of iterations is lower, the algorithm uses the constriction method.

For accuracy, there does not seem to exist a clear trend. However, one of the most interesting tests is that of the function "*Schaffer F6*" in conjunction with the mixed restriction (Figure 3). In this case, most methods tended to suffer from a significant loss in precision (less than or equal to 40% for most). The only method that was able to exceed this small value was the BiLSTM network, with an accuracy percentage of approximately 60%. Although the value is still low, this proves the capability of the BiLSTM network as a robust method in the presence of local minima.

In terms of the variability in the number of iterations to converge, the mixed restriction seems to generate the worst

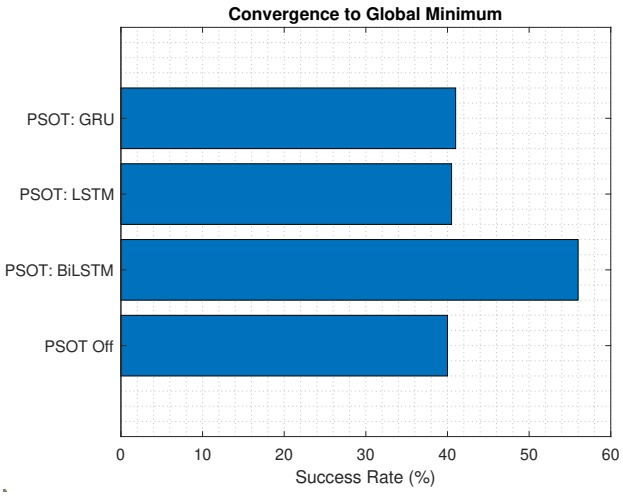

Fig. 3. Convergence precision (convergence of particles towards global minimum) of the PSO algorithm with and without the PSO Tuner (PSOT) enabled, using the "*Schaffer F6*" cost function and "Mixed" restriction for the MPSO algorithm.

results. However, these results are misleading. This high variability comes from the fact that the simulated MPSO algorithm employed used a stopping criteria based on its proximity to the global minimum.

Since the mixed restriction method is characterized by an accelerated convergence towards the first detected minimum, the number of iterations to converge in an environment with many local minimums may consist of either very low (when converging on the global minimum) or very high (when converging on a local minimum) values. Having only 2 types of values, the box-and-whiskers diagram extends the limits of the box from the lowest value (convergence-to-global-minimum iterations) to the highest (number of maximum iterations). The exception to this is the cost function "*Sphere*".

Because it consists of an "easy" function to minimize (evidenced by the fact that all methods were able to find the goal 100% of the time), the only values that were obtained were the values of convergence to the goal. The few exceptions where there was no convergence are marked as outliers.

C. Computation Time

On average, the BiLSTM network seems to be the best choice in terms of its predictive speed, followed by the LSTM network and finally the GRU network. The BiLSTM network presents a lower average prediction time (between 0.4 and 1 ms) in the cases that make use of the inertia and constriction restriction methods (Figure 4).

Initially, the advantage of the BiLSTM network over the remaining networks may seem contradictory. After all, neurons of this type have a greater number of internal parameters and therefore require a greater number of operations to carry out the network's *feed forward* process. However, it should be considered that the final BiLSTM network architecture not only has a lower number of layers (6 in total) but also a lower

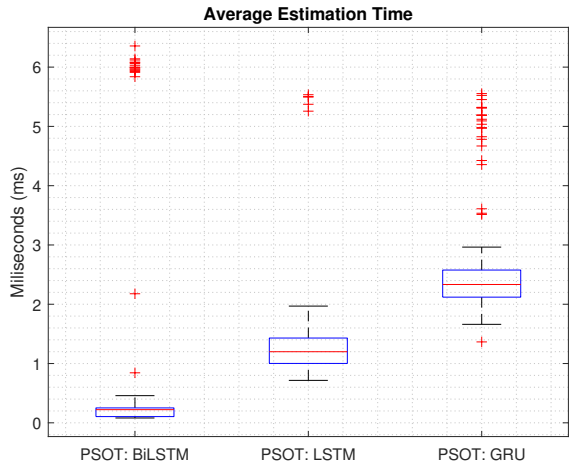

Fig. 4. Computation time for each of the neuron types used for the PSO Tuner.

number of neurons in each of them. Therefore, in the case of the observed results, the reduced number of neurons was more influential than the internal complexity of each neuron.

In turn, it should also be mentioned, that in the case of LSTM and GRU networks, the use of 2 samples is required to obtain optimal results. This leads to additional processing, since the input and output matrices of these networks have a greater dimensionality. Despite all this, in cases where a mixed restriction method is used, the prediction time seems to lack a clear trend.

D. Reduced Particle Number

To test the robustness of the PSO Tuner, it was decided to perform a total of 200 simulations reducing the number of particles to 10. This significantly limits the information available for the MPSO algorithm, causing it to become highly sensitive to local minima. In order to make the test run under the worst case scenario, it was decided to minimize the "*Griewank*" function (a function characterized by a high number of local minima) in conjunction with the inertial and constriction restriction methods. The mixed method was avoided, as it would only enhance the fast convergence behavior towards local minima.

In the case of the constriction method, the MPSO algorithm assisted by the BiLSTM network produced the best results, reaching the goal almost 80% of the time. The LSTM network and the default MPSO were probably able to converge to the global minimum a small number of times thanks to a favorable initial positioning of their particles. The GRU network on the other hand, was never able to reach the global minimum.

When analyzing the motion that produced these results, it can be seen that a decrease in the number of particles brings about an increase in the dispersion of the three types of PSO Tuner. This is a clear indicator of the behavior used by the different neural networks to reach the goal: in the face of a lack of information (caused by the reduced number of particles) the swarm increases its dispersion to cover a greater part of the search region. The maintenance of such dispersion throughout the movement is in turn an indicator of the "hesitant" behavior of the networks, which avoid causing convergence to avoid local minima.

Another important feature is that the movement of the center of the swarms controlled by the LSTM and GRU networks is much more noisy. This could have caused the swarms to orbit around the target, but never converge, proving that they are probably capable of reaching the target, but require a greater number of iterations to do so.

In the case of the inertial restriction, all methods maintained their previous performance, except for the BiLSTM network. Although it continues to position itself as the most accurate option, its accuracy decreased from 80% to 18%. Therefore, if convergence in low-particle swarms is to be improved, the BiLSTM-assisted PSO Tuner and constriction restriction should be used.

V. Discusion

In their base behavior (behavior that is displayed no matter the type of restriction chosen) all networks cause particles to explore the search region by moving together (high coherence) while maintaining a nearly constant swarm diameter. When they find a potential minimum, the particles begin to agglomerate in an attempt to converge. However, if the minimum does not consist of the global minimum (detected by the distance metric between the global best and global minimum), the particles suddenly increase their dispersion to try to escape from the local minimum to which they have converged. If they manage to escape from the previous local minimum, the dispersion increases again and the exploration process starts again.

When this base behavior is exposed to every type of restriction, the particle movement mutates to match the characteristics of the type of restriction. In the case of the inertial restriction, the method acquires a greater capacity to escape from local minima. This is because the greater relative motion of the individual particles (high scattering and low coherence) allows it to accelerate to generate sufficient velocity to escape from the minimum encountered.

In the case of the mixed method, the algorithm acquires a high susceptibility to local minima. When the particles find a minimum, they all increase their speed to rush to that point. If the minimum is not correct, the particles try to escape, but due to the strong influence of the mixed restriction method, they lack sufficient speed to get out of the local minimum.

Finally, in the case of the constriction method, an intermediate result is achieved: the particles converge rapidly towards the first minimum they find, but if they realize that this consists of a local minimum, they alter their dispersion to try to escape. Their escape does not occur as easily or immediately as in the case of the inertial restriction, but after a certain number of attempts they succeed.

This behavior can be explained by observing the evolution of the input and output parameters of the network over time (Figure 5).

At the beginning of its operation, the network aggressively increases the value of ϕ_2 while increasing to a lesser extent the value of ω and ϕ_1. Once the particles find the minimum, the network decreases the value of ω slightly to limit dispersion. This in turn generates a gradual (though noisy) decrease in the coherence of the particles, as the center of the swarm remains static. When the coherence reaches a low enough value, the network aggressively decreases the value of ϕ_1, ϕ_2 and ω, thus reducing the noisy movement characteristic of the PSO algorithm. The new values of ϕ_1, ϕ_2, and ω, consist of values very close to 0, but of the three, ϕ_2 tends to have the highest value. Because ϕ_2 regulates the influence of the global memory of the swarm, a higher value of this constant, in conjunction with a less noisy movement, results in a smooth movement towards the previously found minimum.

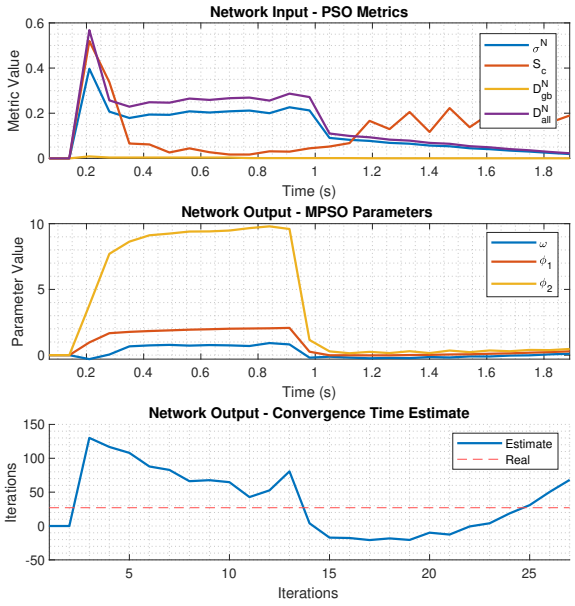

Fig. 5. Evolution of the Inputs and Outputs of the PSO Tuner network.

VI. Conclusion

The PSO Tuner was able to successfully generate automatic and dynamic predictions for the parameters ϕ_1, ϕ_2 and ω of the MPSO algorithm.

The PSO Tuner is able to generate a highly resistant behavior to local minima, thanks to the fact that the trained networks are able to generate parameters that allow the particles to "escape" from the local minima.

All trained architectures displayed an emergent behavior in which they try to imitate the swarm movement seen in the MPSO algorithm while it is subject to different types of restriction. This displays the potential of the PSO Tuner as a mimicking tool, being able to generate behaviors that can cause the standard MPSO to behave as entirely different algorithm.

The PSO Tuner based on the BiLSTM network, presented the best average performance in terms of prediction time, convergence time and accuracy, and smoothing capacity for the swarm movement.

For cases with a low number of particles, the most robust method to increase the accuracy of the standard PSO algorithm is the use of the BiLSTM network, in conjunction with the constriction method.

LSTM and GRU networks require 2 simultaneous samples and the deactivation of the inertial control to produce adequate results.

The GRU network consisted of the network with the worst overall performance in terms of its accuracy and convergence time, as well as the variability inherent to said results.

References

[1] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[2] M. Clerc and J. Kennedy, "The Particle Swarm — Explosion , Stability , and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[3] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *Proceedings of the 2011 3rd World Congress on Nature and Biologically Inspired Computing, NaBIC 2011*, 2011, pp. 633–640.

[4] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," in *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, CSO 2009*, vol. 1, 2009, pp. 505–508.

[5] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization *," in *6th World Congress on Intelligent Control and Automation*, no. 2002, Dalian, China, 2006, pp. 3672–3675.

[6] Y. Feng, G. F. Teng, A. X. Wang, and Y. M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Second International Conference on Innovative Computing, Information and Control, ICICIC 2007*, 2007, pp. 7–10.

[7] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1447, pp. 591–600, 1998.

[8] A. A. Nadalini, L. A. Rivera, and M. Z. Arenales, "Implementation of a PSO trajectory planner using kinematic controllers that ensure smooth differential robot velocities," *IEEE Control Systems Magazine*, 2021.

[9] A. Nadalini, "Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)," 2019.

[10] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 1128–1134, 2008.