Assignment
Swarm Intelligence

1 INTRODUCTION

The primary objective of this assignment is to gain a thorough understanding of the Particle Swarm Optimization (PSO) algorithm, particularly its tuning parameters, through a hands-on approach divided into two parts:

i. Interactive Exploration in NetLogo: This component focuses on utilizing the NetLogo framework to develop an intuitive understanding of how PSO tuning parameters influence the optimization process. Detailed instructions for this part can be found in Section 2.
ii. Application of PSO in Neural Network Optimization: This segment leverages the PySwarm library to replace the traditional backpropagation method for training a Neural Network (NN). In this case, the PSO algorithm will optimize the weights and biases of the neural network, bypassing the usual backpropagation and loss function. Instructions for this part are detailed in Section 3.

The assignment's overall aim is to provide practical experience with PSO and its applications in optimizing complex systems, allowing students to draw connections between theoretical concepts and their practical implementations

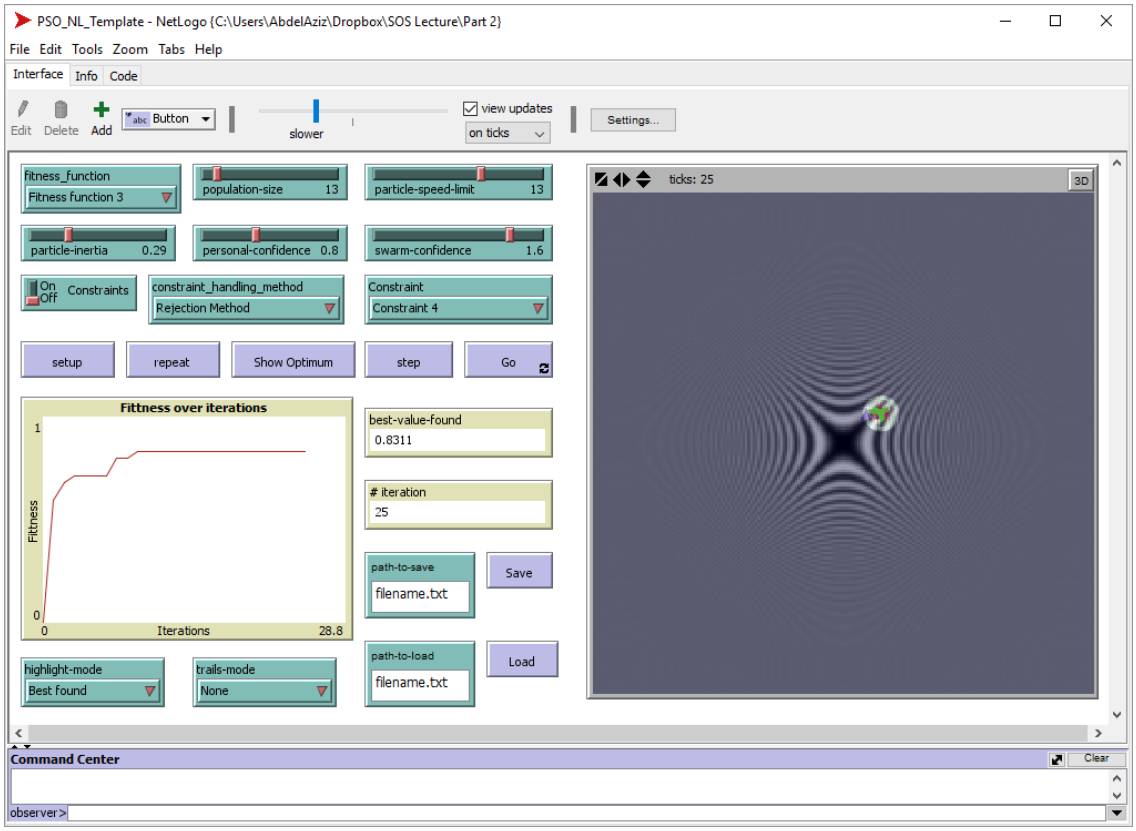
2 NETLOGO INTERACTIVE WORK

In this part of the assignment, you will engage with the NetLogo framework to develop an intuitive understanding of the PSO algorithm. This section does not involve writing code but focuses on exploring the impact of various PSO parameters through interactive experimentation. The primary objective is to grasp the effects of these parameters, which will serve as a foundation for tackling the second part of the assignment.

A comprehensive report documenting your interactive work, experiments, and observations is crucial for completing this assignment. The list of functions and constraints available for experimentation as well as an overview of the NetLogo framework are provided Section 6.

2.1 THE NETLOGO PSO PLAYGROUND (NL-PSO)

The NetLogo PSO Playground (NL-PSO) is a tool that facilitates interactive work with the PSO algorithm, allowing optimization of functions in two dimensions (2D). Although optimizing functions in 2D is relatively straightforward, it offers a clear visualization of the PSO process, promoting a deeper understanding of the algorithm.



Key features of the NL-PSO include:

- ✓ A set of pre-implemented fitness functions and constraints for interactive experimentation, as described in Section 6.
- ✓ A graphical user interface (GUI) for configuring core PSO parameters, selecting fitness functions, and enabling or disabling constraints, etc.
- ✓ Visualization controls to monitor fitness values and iteration counts both numerically and graphically.
- ✓ Options for repeating experiments with consistent initial conditions, along with functionalities for saving and loading experiments. This enables a focused study on how specific parameter changes influence the optimization process.
- ✓ Optional constraint functions that can be activated or deactivated to observe how PSO behaves under different conditions.
- ✓ The algorithm is designed for maximization; hence, fitness values are normalized between 0 and 1, with 1 representing the optimal solution.
- ✓ In cases where multiple optima exist with equal fitness values, the algorithm terminates upon reaching any of them.
- ✓ The framework uses a toroidal wrapping principle, where particles that reach the edge of the search space reappear from the opposite side, akin to a seamless loop.
- ✓ The template uses the toroidal wrapping principle, that is the particles wrap around when they reach a boundary of the search space, e.g. when a bird passes the eastern boundary, it arises from the west again.

2.2 Experiments and Analysis

You are required to conduct experiments using the NL-PSO tool, varying parameters such as swarm size, inertia (*w*), velocity limits, and attraction factors (c1 for personal confidence and *c2*

for social confidence) as well as the swarm size. The aim is to develop an understanding of how these factors influence the algorithm's convergence behaviour. It is crucial to examine the algorithm's convergence regarding:

- ✓ The quality of the solution obtained.
- ✓ Avoiding local minima.
- ✓ Preventing stagnation in relation to specific parameter configurations.

The following are examples what you could try:

- Experiment with different swarm sizes and inertia values to analyze their impact on convergence speed and accuracy.
- Investigate which values of self-confidence relative to swarm-confidence or inertia promote stagnation and/or early convergence and which values avoid it.
- Explore the effects of varying velocity limits on the search process.

Relate the convergence behaviour to different functions chosen for experimentation. This will demonstrate that distinct parameter configurations are required for different problem topologies to achieve optimal convergence.

You should design experiments to highlight the advantages and limitations of various parameter settings across different cases. Document your findings and insights based on empirical results to illustrate how parameter adjustments influence performance. To ensure the robustness of your conclusions, each experiment should be repeated at least 10 times, depending on the variability of the outcomes.

3 PSO-NN Optimization

In this portion of the assignment, you will utilize the PySwarm Python library to optimize a neural network (NN) using the Particle Swarm Optimization (PSO) algorithm. The primary objective is to showcase the capability of the PSO algorithm in optimizing a vast number of continuous variables, demonstrating its strength in searching large solution spaces.

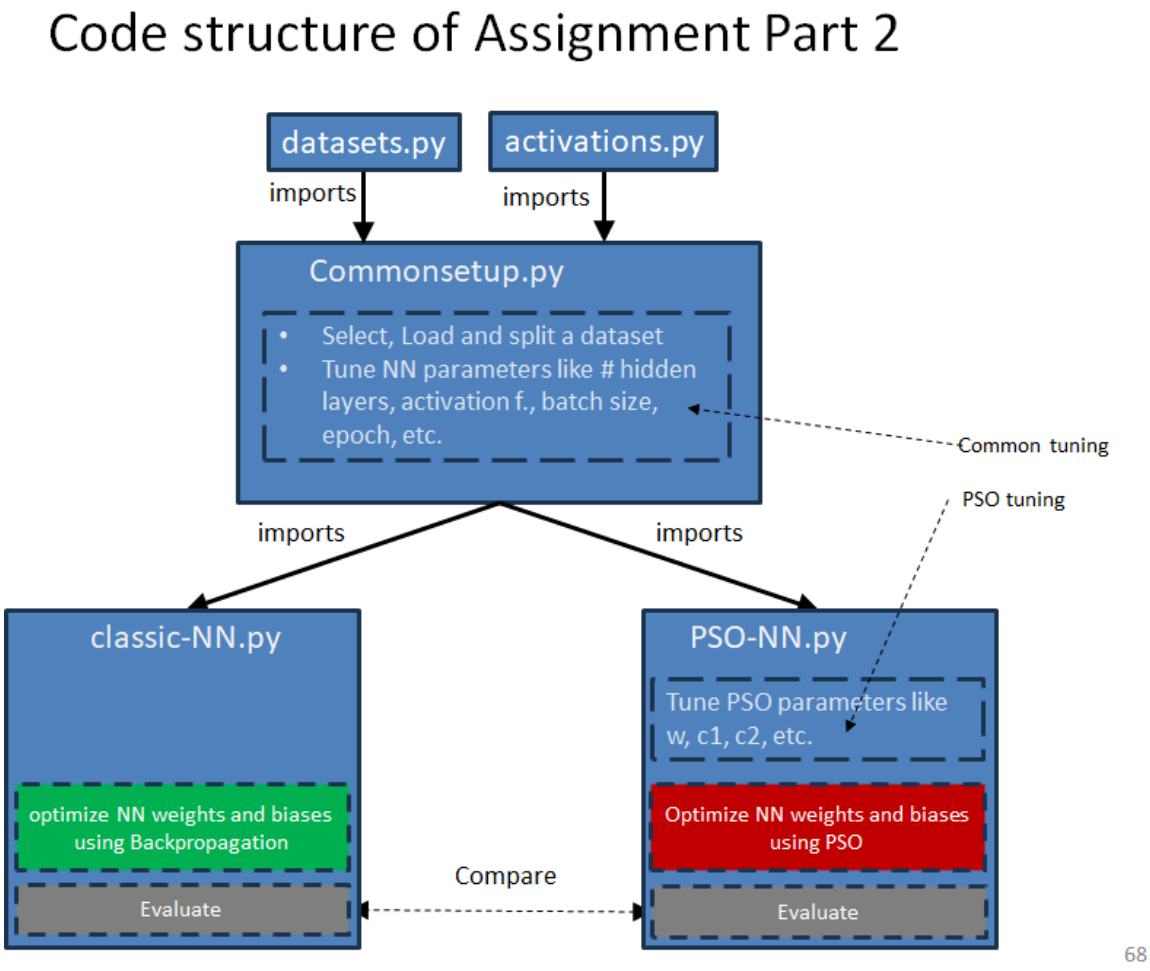
This section aims to achieve the following:

I. Demonstrate the ability of the PSO algorithm to efficiently optimize high-dimensional variables, specifically the weights and biases of neural networks.
II. Develop a deeper understanding of how tuning the PSO parameters (e.g., inertia w, personal confidence c1, and social confidence c2) affects its optimization performance.
III. Introduce PySwarm as a robust tool for implementing PSO algorithms in Python.

3.1 Implementation Details

In this task, you will replace traditional backpropagation with the PSO algorithm to optimize the weights and biases of a neural network. Depending on the dimensions of the chosen network (i.e., the number of input, hidden, and output layers), the number of variables to optimize could be several hundred or more. Proper parameter tuning will be essential to handle this complexity effectively.

The evaluation will involve comparing the performance of the PSO-optimized network with a traditional backpropagation-based network. Both approaches will use the same network architecture and training data, enabling an objective comparison of their performance. The final evaluation will be conducted on a shared test dataset.



3.2 Implementation Overview

You are provided with several Python files to kickstart your implementation. See the figure above:

I. **datasets.py:** Contains functions to download and preprocess datasets into a standardized format. You are required to choose two datasets for your experiments. This file is complete and does not require modification unless you opt to use additional datasets.

II. **activation.py:** Defines various activation functions and their derivatives, which can be used for tuning your neural network. While this file is also complete, you may extend it with additional activation functions if needed.

III. **commonsettings.py:** This file sets up the neural network and manages data loading and splitting. Note that the settings in this file apply to both the traditional neural network and the PSO-optimized version, ensuring consistent comparisons.

IV. **classic-NN.py:** Implements a fully functional neural network using standard backpropagation for training. This file is ready to use without modifications.

V. **PSO-NN.py:** Implements the same neural network architecture as in classic-NN.py, but replaces backpropagation with PSO optimization. **A minor section of this file, specifically the fitness function, needs to be completed** by you.

3.3 Fitness Function Completion

As part of the assignment, you will need to complete the implementation of the fitness function in the PSO-NN.py file. This requirement ensures that you understand the underlying algorithm. The concept of the fitness function is straightforward: each solution (i.e., the coordinates of a particle) represents a set of weights and biases applied to the neural network. The fitness function then evaluates the accuracy of the network on a batch of test instances, with the resulting accuracy serving as the fitness value.

To implement this function correctly, follow the guidelines and comments embedded in the code. The implementation should be concise, ideally within 2 to 10 lines of code. If your solution exceeds 10 lines, it might indicate an overly complex or wrong approach.

Note: Upon downloading the provided files, the traditional neural network (classic-NN.py) should function immediately by executing *python classic-NN.py*. The PSO-based network (PSO-NN.py) will also run without errors, but its current fitness function returns random values, resulting in poor accuracy. This is the part you need to correct to achieve meaningful results.

3.4 Experiments and Analysis

You are expected to conduct experiments to evaluate the effectiveness of the PSO algorithm in optimizing the neural network. The goals include:

- ✓ Comparing the performance of PSO-based optimization with traditional backpropagation in terms of accuracy and convergence speed.
- ✓ Comparing behaviour and results in three different datasets your choice. You are allowed to choose any, but you are expected to reason, explain and compare the results.
- ✓ Analyzing the influence of various PSO parameters (w, c1, c2, and velocity limits) on optimization performance.
- ✓ Identifying strengths and weaknesses of PSO for neural network training, particularly in terms of handling high-dimensional search spaces.

For each experiment, summarize your approach, objectives, and observations. Focus on presenting aggregate results and significant trends rather than individual data points. Where relevant, include representative plots to illustrate the impact of parameter variations on the fitness and convergence behaviour.

4 Reporting Guidelines

For this assignment, you are required to report your work, which you will submit to TUWEL.

The assignment consists of two distinct parts: interactive exploration using the NetLogo framework and the PySwarm implementation for optimizing a neural network. You have the flexibility to decide how to present your findings. You can choose to:

- Submit two separate reports: one for the NetLogo component and another for the PySwarm component.
- Alternatively, you may integrate both parts into a single comprehensive report.

Regardless of the format you choose, it is essential to maintain a clear and coherent report structure. Each section should be clearly delineated, and the content should remain logically organized. Ensure that your report—whether combined or separate—includes all the required sections outlined below.

4.1 Abstract

Provide a brief overview of the key objectives of your assignment. This section should succinctly summarize the problem you tackled, the approach you employed, and the main findings of your work.

4.2 Implementation

This is only for the PSO-NN part,since the NetLogo part does not have implementation.

- Describe your implementation of the PSO algorithm without including any code. Clearly outline the choices you made, including parameter selection, and provide the rationale behind these decisions.
- Offer a high-level description of your implementation. For instance, if you used specific techniques for exception handling, briefly explain the purpose and how it was achieved.
- Document any challenges encountered and how you resolved them, along with observations made during the implementation process.

4.3 Experiments

- Clearly detail the experiments you performed, including the objectives and the reasoning behind your choices. Relate these experiments to the concepts discussed in the lectures.
- Provide a justification for your parameter selections and explain how these settings align with your experimental goals.

4.4 Results and Analysis

- Summarize the outcomes of your experiments, focusing on aggregated results rather than individual data points. Include averages or summaries for each experiment and case.
- Present representative plots (e.g., fitness vs. iterations) where applicable to support your analysis.
- Interpret your findings by discussing how different parameters influenced the performance and convergence behavior of the PSO algorithm. Highlight the advantages and disadvantages of the PSO approach based on your observations.

4.5 Conclusion

- Conclude your report with an overall summary of your findings. Reflect on the performance of your PSO implementation compared to traditional backpropagation.
- Identify the strengths and weaknesses of the PSO approach, particularly in the context of the specific cases you tested.

5 SUBMISSION GUIDELINES

This assignment should be completed in groups of up to three students. While it is possible to work individually or in smaller groups, doing so will not reduce the workload or affect grading criteria.

Please follow these submission instructions:

5.1 SUBMISSION FORMAT

Upload a single compressed package (in .zip, .tgz, or .rar format) to the TUWEL platform. Ensure the package is named as follows:

SOS2024_Ex2_group<group_number>_<studentID1>_<studentID2>.zip

This package must include:

1. A PDF file Report.pdf containing your report according to the description above (or two files if you choose to report the two parts each separately)
2. A folder containing three folders each corresponding to one of the selected datasets and having the five python files after completing the fitness function and tuning the PSO for the dataset. They should be runnable straightforwardly and reflect the reported results.

5.2 DEADLINE

The assignment is to be submitted no later than **SO 04.01.2026 23:59.**

5.3 OPTIONAL FEEDBACK

You are encouraged to provide feedback on the assignment, indicating which sections were beneficial and which were less helpful. Suggestions for additional experiments or improvements are welcome. This feedback can be submitted either anonymously via the TISS feedback mechanism or included in your report. Note that providing feedback is entirely optional and will not influence your grade.

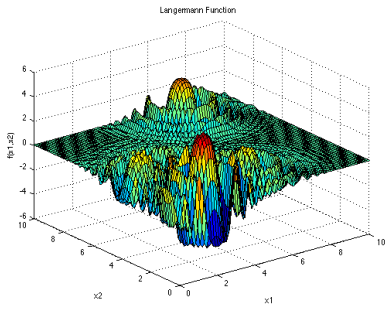
6 APPENDIX

6.1 FITNESS FUNCTIONS

The assignment includes a selection of test functions for optimizing with the PSO algorithm.

1. *LANGERMANN FUNCTION*

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \exp\left(-\frac{1}{\pi}\sum_{j=1}^{d}(x_j - A_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{d}(x_j - A_{ij})^2\right)$$



2. *Schwefel function*

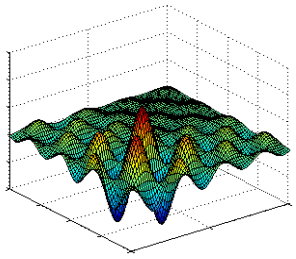
$$f(x_1, \dots, x_n) = \sum_{i=1}^{n}\left(-x_i \sin\left(\sqrt{|x_i|}\right)\right) + \alpha \cdot n$$

$with\ \alpha =$ 418.9829 and $-512 < x_i < 512$

3. *Shubert function*

$$f(x,y) = \left(\sum_{i=1}^{5} i\cos((i+1)*x+i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)*y+i)\right)$$

$with -100 < \mathrm{x,y} < 100$

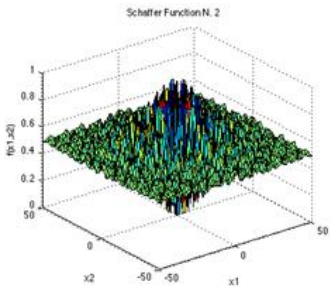


Shubert function

4. *Schaffer function*

$$f(x,y) = 0.5 + \frac{sin(x^2 - y^2)^2 - 0.5}{(1 + 0.001 * (x^2 + y^2))^2}$$

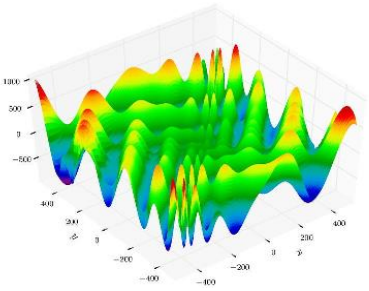
$with -100 < \mathrm{x,y} < 100$



5. *Eggholder function*

$$f(x,y) = -(y + 47)\ \sin\left(\sqrt{\left|\frac{x}{2} + (y+47)\right|}\right) - \mathrm{x}\sin\left(\sqrt{|x-(y+47)|}\right)$$

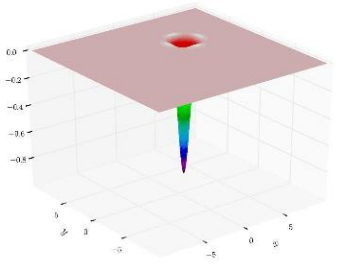
$with\ -51200 < x,y < 51200$



6. Easom function

$$f(x,y) = \cos(x)\cos(y)\ e^{\left(-((x-\Pi)^2+(y-\Pi)^2)\right)}$$

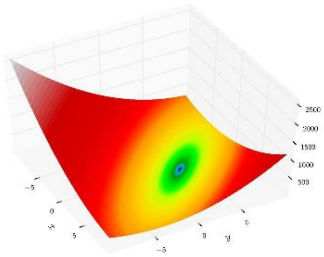
Since the implementation in the template is maximum-based, implement a negative variant of this function to have the sharp peak as an optimum.



Easom function

7. Booth's function

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2$$



Booth's function

6.2 CONSTRAINTS

The following list of constrains is available in the NL-PSO:

- $c1(x,y): x^2 + y^2 < 6000$
- $c2(x,y): x > 3$ y or $3x < y$
- $c3(x,y): x > y + 20\ or\ x < y - 20$
- $c4(x,y): x^2 + y^2 < 9000\ and\ x^2 + y^2 > 4000$
- $c5(x,y): x > y$
- $c6(x,y): 10x < y^2$
- $c7(x,y): \tan(2x) < \tan(4y)$
- $c8(x,y): \sin(8x) < \sin(8y)$
- $c9(x,y): \sin(x) * \sin(y) < 0.2$
- $c10(x,y): \tan(x\ y) < 1$

6.3 THE NETLOGO FRAMEWORK

At its core, NetLogo uses a grid-based environment where individual cells, known as patches, represent the landscape, and agents, referred to as turtles, interact with this environment. The framework supports both a graphical user interface (GUI) and a programmable interface, allowing users to define how agents interact with their surroundings over time. The progression of time within the simulation is managed by discrete time steps, known as ticks, which drive the simulation forward.

NetLogo provides extensive capabilities for visualizing processes in both 2D and 3D, enabling users to observe how agents interact with the environment dynamically. The framework is renowned for its ease of setup and use, featuring a straightforward scripting language that is both accessible and efficient. This simplicity, coupled with its flexibility, has made NetLogo a popular tool in educational settings, with a large global community of students and educators.

In addition to its core functionalities, NetLogo includes a comprehensive library of pre-built models that cover a wide range of domains. This library is an excellent starting point for users to explore existing simulations and gain insights into different system behaviors. For those new to NetLogo, numerous tutorials are available online, including the highly recommended series by Gabriel Wurzer (Lessons 1 to 17).

To begin using NetLogo, download the framework from the official website, install it on your system, and explore the provided models. Engaging with tutorials and experimenting with the library will help you quickly become proficient in using this powerful simulation tool.