Assignment 3
Swarm Intelligence

1 GOAL OF THE ASSIGNMENT

The goal of this assignment is that the students understand the Particle Swarm Optimization (PSO) algorithm by (i) implementing variants of the PSO algorithm using the NetLogo Framework, (ii) visually observing the process of the PSO algorithm, (iii) performing specific experiments on these algorithms that help understanding the core aspects of the PSO.

2 THE ASSIGNMENT

This assignment is to be done in groups up to 3 students. Doing it in smaller groups or individually is basically allowed but does not reduce the requirement or give advantages in the grading.

The assignment is based on the NetLogo framework (a short overview is provided in Section The NetLogo Framework). For the assignment, implement a PSO algorithm with extensions based on a template that will be provided. This template is a basic PSO algorithm that has some functionalities to be completed, changed or extended (More information about this template is provided in Section The NetLogo Template). Choose a combination of options as described in Section Options. Once implemented, perform experiments on your implementation, analyse those results, and report your work.

1. Download the template from TUWEL. Play with it by modifying the code and running sessions until you understand it, such that you can modify its functionalities according the requirements.
2. Select with your group members an algorithm variant by choosing from the combinations described in Section Options2.1. Implement this algorithm variant carefully according to the description.
3. Perform tests on the algorithm you have implemented according to the instructions in Section Experiments.
4. Report your implementation and the results of your tests according to the description in Section Report.
5. Upload your implementation and report before the submission deadline. Be sure that your submission meets the submission guidelines in Section Submission Guidelines.
6. (optional) Provide feedback on the exercise in general: which parts were useful / less useful; which other kind of experiment would have been interesting, … (this section is, obviously, optional and will not be considered for grading. You may also decide to provide that kind of feedback anonymously via the feedback mechanism in TISS – in any case we would appreciate learning about it to adjust the exercises for next year.)

2.1 OPTIONS

The PSO algorithm in the template already supports constraint handling based on the **Rejection** method. In your implementation, extend the algorithm to another constraint handling method based on the **penalty** method. Implement different fitness functions and different constraints. For this, select a combination of

- ✓ At least three fitness functions from the fitness function list below and
- ✓ At least three sconstraints from the constraint list below.

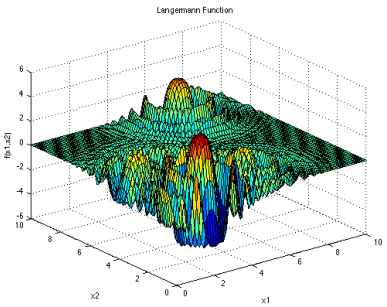
2.1.1 Fitness functions

Choose two or more different fitness functions from the list below or other test functions from this link . Implement the two functions, such that they are selectable via the "fitness function" control in the template.

The following functions are selected test functions that should simulate the fitness functions to be optimized by the PSO algorithm. For more information about these test functions (e.g. domains, ranges, etc.), please refer to this link . Note that some functions are defined for arbitrary number of variables. Since we deal in this assignment only with two variables (coordinates), namely x and y, adapt the definitions for this case. Also, note that the corresponding domains may have to be scaled to fit in the ranges of the coordinate system of NetLogo (one example function is already implemented in the template, which also shows how to scale if needed).

1. *LANGERMANN FUNCTION*

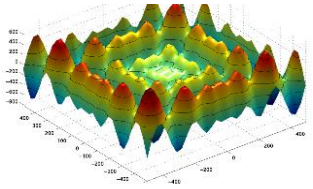
$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i \exp\left(-\frac{1}{\pi}\sum_{j=1}^{d}(x_j - A_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{d}(x_j - A_{ij})^2\right)$$



2. *Schwefel function*

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

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

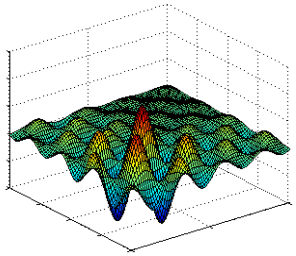


Schwefel Function

3. *Shubert function*

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

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

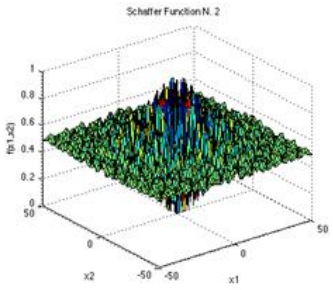


Shubert function

4. Schaffer function

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

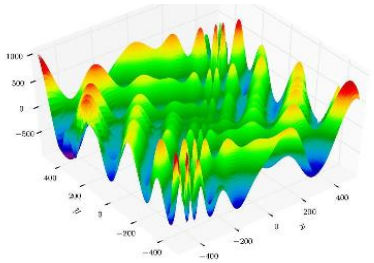
$$with -100 < x, y < 100$$



5. Eggholder function

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

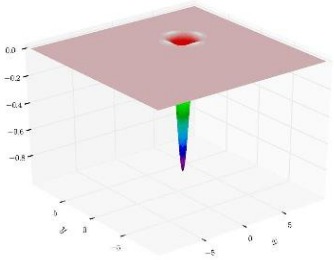
$$with\ -51200 < x, y < 51200$$



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.

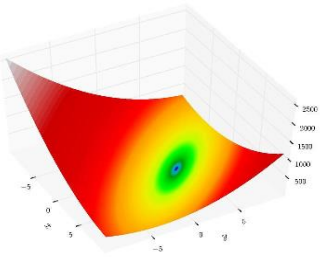


Easom function

7. Booth's function

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

Since the implementation in the template is maximum-based, you may choose to implement a negative variant of the function to avoid that the optimum is at the edge of the search space.



Booth's function

2.1.2 Constraints

In the template a constraint handling method based on rejection is already implemented. Implement another constraint handling method based on penalty, as has been described in the lecture. Use the option control "Constraint handling method" to switch between them. Also, use the switch control "Use constraints" to determine, whether constraints are used or not. You will need these options for your experiments.

The following list of constrains is available:

- $c1(x,y): x^2 + y^2 < 6000$
- $c2(x,y): x > 3\,\mathrm{y}$ or $3\mathrm{x} < \mathrm{y}$
- $c3(x,y): x > y + 20\ or\ x < y - 20$
- $c4(x,y): x^2 + y^2 < 9000\ and\ x^2 + y^2 > 4000$
- $c5(x,y): x > y$

- $c6(x,y)$: $10x < y^2$
- $c7(x,y)$: $\tan(2x) < \tan(4y)$
- $c8(x,y)$: $\sin(8x) < \sin(8y)$
- $c9(x,y)$: $\sin(x) * \sin(y) < 0.2$
- $c10(x,y)$: $\tan(x\ y) < 1$

2.2 Experiments and Analysis

Perform experiments using the implemented algorithm. Run different sessions with different fitness functions and try to vary the parameters (swarm size, inertia, velocity limits, attraction factors, etc.), such that you have a feeling of the influence of these factors on the progress of the algorithm. Note that the particles are randomly initialized each time the setup is performed. If you want to repeat an experiment with the same initialization, you can do this by using the "repeat" button or by using the "save" and "load" buttons. Note that the experiment is saved automatically in a file "backup.txt" each time the setup is called (note that this file should be copied somewhere else, otherwise it is overwritten).

Design and perform experiments considering:

(1) The convergence of the algorithm regarding (i) the quality of the solution found, (ii) sticking in local minima, (ii) stagnation in relation to algorithm parameters, namely
   a. Swarm size
   b. Inertia w
   c. The attraction factors c1 and c2 (self-confidence and swarm-confidence)
   d. Velocity upper limit

(2) Relate the convergence aspects in (1) to the different functions you have, i.e. to show for example that for functions with different topologies, different parameters are required for an efficient and performant computation.

(3) Design experiments that demonstrate the differences (advantages and disadvantages) between the two constrains handling methods, namely the rejection and the penalty and discuss these.

(4) Perform diverse experiments, i.e. with different combinations of with/without constraint, with different constraints, and with different fitness functions.

Analyse and report your experiment results. Use the empirical results to show how the aspects and relations above.

We define optimal convergence, early convergence, and stagnation as follows:

- Optimal convergence: The swarm reaches the global optimum, i.e. a point with fitness value of 1 (the algorithm stops automatically).
- Early convergence: all birds are at the same point and don`t change their positions, but the fitness value of the swarm is < 1.
- Stagnation: no progress in the swarm fitness value, where in contrast to early convergence, some of the birds are still changing their position, which theoretically makes it possible that the fitness value changes.

In order that your conclusions are robust, perform at least 10 to 30 times for each experiment., depending on the divergence of the observations.

2.3 Report

Provide a report that summarizes your work which contains the following parts:

Abstract: a paragraph describing of the main concern of your work.

Implementation: A description of your implementation of the PSO algorithm. Please don't include any code in the report. State clearly, which options you have selected and the reasons for your selection. Provide a high-level description of your implementation. E.g. if you describe the exception handling, describe shortly your aim and how your realized it. Write your notes and comments on your observations and eventually problems you faced and how you solved them.

Experiments: Describe clearly the experiments you have performed. Provide the goal of each experiment and a justification for the experiment choice in relation to your options as well as the topics covered in the lecture.

Results and analysis: Summarize your analysis. You do not need to report all experiment instances, but rather summaries and averages of each experiment, case, etc. You may include presentative plots of fitness/iterations in your report. Try to consider the relations and observation to make conclusions on the performance, advantages, and disadvantages of the PSO variants, thereby taking into consideration the specific options you have selected, that is how the fitness function, the constraint and the constraint handling method are related to the performance and convergence behaviour.

2.3.1.1 Conclusion

Conclude your work by providing an overall summary of your work. Here you relate the performance of your algorithm to the options you have selected and provide strength and weakness of PSO in general and in relation to your specific cases.

2.4 Submission Guidelines

Upload one package file in zip/tgz/rar format to TUWEL that contains all your files not later than the deadline below. This package should be named as OS2016_Ex3_group<groupno>_<studentID1>_<studentID2>.zip Your submission should contain:

1. A runnable NetLogo file named SOS2016_Ex3_group<groupno>_<studentID1>_<studentID2>_algorithm.nlogo, which contains your implementation.
2. A PDF file called SOS2016_Ex3_group<groupno>_<studentID1>_<studentID2>_report.pdf according to the description above

Submission deadline: 31.12.2019

3 The NetLogo Framework

The NetLogo Framework is a powerful simulation framework specialised on physical and biological systems that evolve with time (e.g. self-organizing systems).

NetLogo provides a modelling instrument for ecosystems that mainly consists of (i) a grid whose cells are called patches, which should model the environment and (ii) individuals called turtles, which should model the agents interacting with the environment. NetLogo provides a GUI interface and a programmable interface that enable the user to define how the agents interact

with the environment over time. The time is modelled by a takt, called tick, which drives the simulation process.

NetLogo visualizes process in 2D and 3D to show visually how the agents interact with the environment over time.

NetLogo is easy to setup and implement. It uses a simple language with a syntax that is easy to learn. NetLogo has a large community and is used worldwide by thousands of students and teachers.

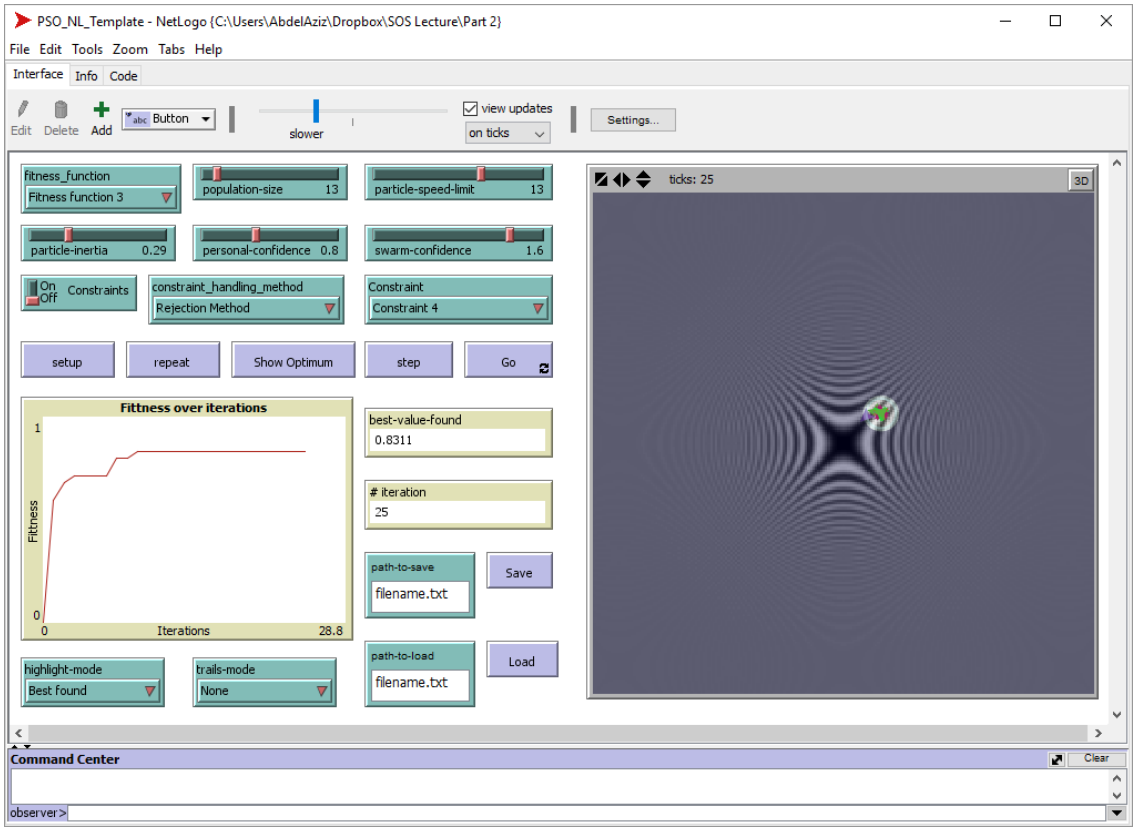
NetLogo provides a library of numerous already implemented models. This library can be found here: NetLogo library. There are many tutorials in the web. The tutorial by Gabriel Wurzer, Lessons 1 to 17 is recommended.

To get familiar with NetLogo, download the framework from and install it on your desktop. Watch a tutorial and start with downloading models from the library and playing with them.

4 The NetLogo Template

The template PSO_NL_Template.nlogo can be used as a starting point for implementing the PSO algorithm. It contains a complete implementation of a basic PSO algorithm with constraint handling based on rejection method. The following are notes about this template.

1. The implementation assumes a maximization optimization.
2. All fitness values calculated from a fitness function are normalized using a min-max normalization, i.e. are between zero and one where one is always the optimum.
3. Some functions have more than one optimum with equal fitness values of 1. The algorithm stops, once it reaches one of them.
4. 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.
5. The template contains a GUI interface for
    - setting the basic parameters of the PSO algorithm
    - choosing a fitness function
    - choosing a constraint handling method
    - choosing a constraint
    - switching the constraint on and off
    - other functionalities to control the process visualization
6. It provides controls for repeating experiments with same initialization, but with different parameter as well as saving and loading experiments
7. It contains controls for showing the actual fitness value and number of iterations
8. In the implementation, there are incomplete functions e.g. dummy fitness functions or constraint functions that should be implemented by the students.


PSO_NL_Template - NetLogo (C:\Users\AbdelAziz\Dropbox\SOS Lecture\Part 2)
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit Delete Add Button slower view updates on ticks Settings...
fitness_function
Fitness function 3
population-size 13
particle-speed-limit 13
particle-inertia 0.29
personal-confidence 0.8
swarm-confidence 1.6
On Off Constraints
constraint_handling_method
Rejection Method
Constraint
Constraint 4
setup repeat Show Optimum step Go
Fittness over iterations
Fitness
Iterations
best-value-found
0.8311
iteration
25
path-to-save
filename.txt
Save
path-to-load
filename.txt
Load
highlight-mode
Best found
trails-mode
None
ticks: 25
3D
Command Center
Clear
observer>