FLWSwarm
Follow the Leader + Random Walk Swarm Algorithm

A tool for numeric optimisation of bound-constraint cost functions using agent-based simulation

User Guide

Version 1.5

Sergio A. Rojas, PhD.

Universidad Distrital Francisco José de Caldas

Bogotá, Colombia, May 2022

FLWSwarm Version 1.5 - User guide

Copyright © 2022 Sergio A. Rojas

This document is distributed under the CC BY-NC-ND license (*Creative Commons Attribution-Noncommercial-NoDerivatives 3.0*). Any other unauthorised form of distribution, copying, duplication, reproduction, or sale (total or partial) of the content of this document, both for personal and commercial use, will constitute an infringement of copyright. This guide is an original work of its author, and therefore it is protected by the laws that regulate copyright and intellectual property. The opinions and points of view expressed in this document are personal to the author and do not compromise the policies, intentions, strategies, or official position of any other organism, company, organisation, service or person mentioned in it.

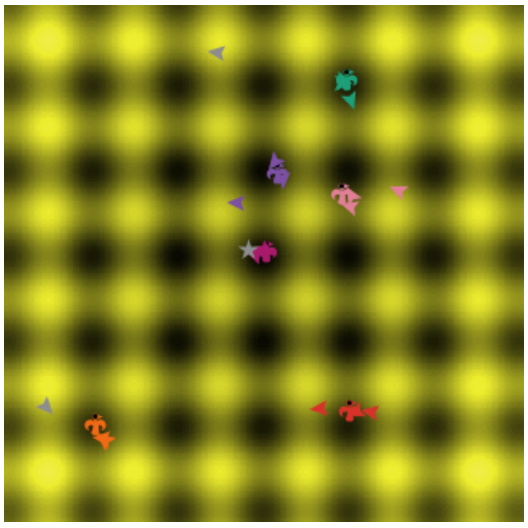
The author has made every effort to ensure that this guide is free from errors or omissions. However, the author accepts no responsibility for offence, damage or loss caused to any person acting or endorsing actions using the material contained in this document.

First Edition, May 2022
Bogotá, Colombia

Overview

FLWSwarm is a software tool designed to find approximate solutions to optimisation problems whose decision variables take numeric values in the real domain. It is based on an agent-based model that implements a swarm algorithm using mainly two search operators: Follow the Leader and random Walk. The goal of the swarm is to discover the best patch within a discretised landscape for a given optimisation problem, that is, to find the optimal (minimum) value of the quantised cost function for said problem, in a more efficient way than traversing the entire landscape (exhaustive search).

The approach of solving an optimisation problem with a swarm of artificial agents undergoing an adaptation process is known as Swarm Intelligence algorithm. Instead of using mathematical analysis of aggregated variables describing the phenomena, this approach resorts to modelling the interaction of a group of individuals in a simulated environment and trace the evolution of such variables as the simulation progresses. In this way, FLWSwarm assumes that global information about the problem emerges as an intrinsic property of the evolution of the algorithm, which can not be explained away by isolated contributions of single agents. In addition, visual inspection of the emerging patterns of agents self-organisation, in response to changes in the simulation parameters, can give useful insights regarding the hidden particularities of the problem.



FLWSwarm v1.5 has been released under GNU General Public License (GPLv3); it is available online at:

http://modelingcommons.org/browse/one_model/6978

Contents

Overview iii

1 Description of the tool 1
- 1.1 What is FLWSwarm? . . . 1
- 1.2 How it works . . . 2
- 1.3 How to use it . . . 3
- 1.4 Other distinctive features . . . 6
- 1.5 Try it yourself . . . 8
- 1.6 Extending the tool . . . 9

2 Installation and execution 11
- 2.1 Online version . . . 11
- 2.2 Desktop version . . . 12

3 Source code 15

4 Software license 23

Chapter 1

Description of the tool

1.1 What is FLWSwarm?

FLWSwarm is an agent-based model that implements a swarm search algorithm for numerical optimisation of real-valued bound-constraint functions, based mainly on two search operators: Follow the Leader and Random Walk (FLW). The goal of the swarm is to discover the best patch within a discretised landscape for a given optimisation problem, that is, to find the optimal (minimum) value of the discretised cost function for said problem, in a more efficient way than traversing the entire landscape (exhaustive search). The complete set of operations performed by the algorithm are:

- Follow the leader: there are groups of agents that follow a leader.
- Random walk: there are scout agents moving on their own.
- Single- or multi-leaders mode: there can be multiple leaders in the swarm.
- Leadership relay: leadership is constantly updated based on fitness.
- Clash safeguard: leader agents must stay away from each other to avoid collisions.
- Monitor premature convergence: when cohesion is too low, agents disperse.
- Enable elitism: an agent stays near the best patch discovered so far.

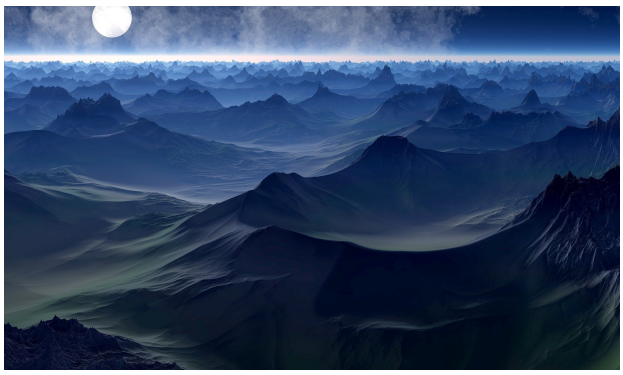


Figure 1.1: FLWSwarm is a tool to search for the highest (lowest) peak of a simulated landscape generated from the cost function of an optimisation problem.

1.2 How it works

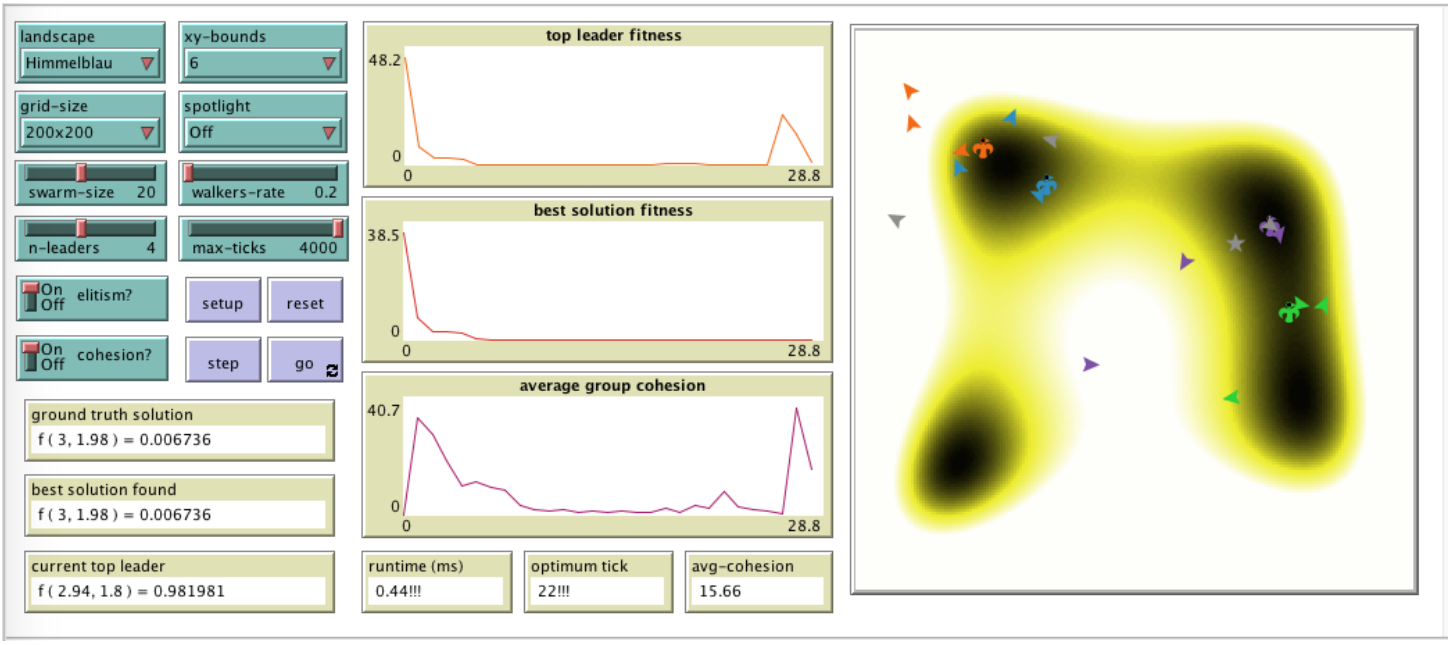


Figure 1.2: FLWSwarm user interface.

The optimisation problem defined by the LANDSCAPE chooser (from SPHERE, RASTRIGIN, ROSENBROCK, HIMMELBLAU, EGGHOLDER, or RANDOM), is discretised and projected onto the 2D grid of cells (patches) that make up the view area of the simulation. Therefore, the swarm of agents aims to discover the optimal patch in said projected landscape, that is, the coordinates where the cost function is the lowest.

Each agent is characterised by a location *(x,y)* in the landscape and the value of the cost function evaluated at that patch, which also indicates its fitness as a candidate solution for the chosen problem. Now each agent can have one of three possible roles (or breeds): LEADER (an agent located in a promising patch), FOLLOWER (an agent following a leader), or WALKER (a scout agent).

In the simulation, LEADERS are shown as a "bird" shape (reminiscent of earlier models of swarm intelligence inspired by the behaviour of flocks of birds) and a different color, while FOLLOWERS are shown as a triangle and the color of his LEADER. WALKERs, in turn, are shown as a gray "triangle" shape. As a final solution to the problem, the algorithm maintains a FLW-BEST-PATCH ever discovered during the simulation run. Similarly, the TOP-LEADER represents the best solution during a given iteration.

LEADERs and WALKERs move randomly around their current positions, however LEADERs move within a small range of distance (local exploitation) while WALKERS move within a larger range (global exploration). When a WALKER accidentally discovers a more promising patch than those of the current leaders, the TOP-LEADER is asked to move to that patch.

Furthermore, every LEADER has an (equally larger) group of FOLLOWERs. FOLLOWERs move in pursuit of their leaders: they face their corresponding leader and jump in that direction with a random step length. When a FOLLOWER accidentally discovers a more promising patch, he switches roles to become the LEADER of their group (in practice, since all agents are homogeneous, they simply trade places).

In each iteration of the algorithm, all the agents of the swarm traverse the landscape moving according to their roles. Additionally, three operators were introduced to improve search efficacy:

- *Monitor premature convergence*: When the average cohesion of groups within the swarm falls below a certain level, agents disperse by performing a warm restart to random locations.
- *Clash safeguard*: Prevents two leaders with their groups from colliding in the same landscape region.
- *Elitism*: Recalls the best discovered patch (the best solution found so far) by moving a WALKER to that position.

The simulation ends after a maximum number of steps (MAX-TICKS) is reached, or when the GROUND-TRUTH-SOLUTION is discovered first.

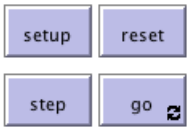
1.3 How to use it

Firstly, from the control panel shown above, choose an optimisation problem to solve from the LANDSCAPE drop-down list and select the corresponding XY-BOUNDS (recommended values are [-6, 6] for SPHERE, RASTRIGIN, ROSENBROCK, and HIMMELBLAU, and [-512, 512] for EGGHOLDER and RANDOM). The GRID-SIZE chooser allows defining the sampling resolution of the 2D grid of patches where the cost function will be evaluated (the larger this size, the harder it is to discover the best patch).

Then choose the number of agents to simulate with the SWARM-SIZE slider, the percentage of walkers in the swarm with the WALKERS-RATE slider, and the number of N-LEADERS within the swarm. Also, choose the termination condition with the MAX-TICKS slider and use the ELITISM? and COHESION? switches to enable or disable the corresponding operators. A typical configuration of these parameters would be (to simulate a population of 20 agents, including 4 leaders with 3 followers each, plus 4 walkers):

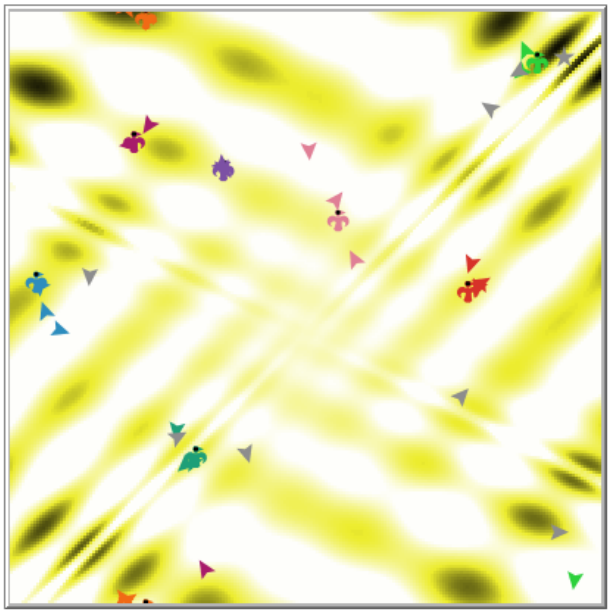
- SWARM-SIZE: 20
- N-LEADERS: 4
- WALKERS-RATE: 0.2
- ELITISM?: On
- COHESION?: On
- MAX-TICKS: 1000

Now you can run the simulation using the buttons in the control panel (shown below):

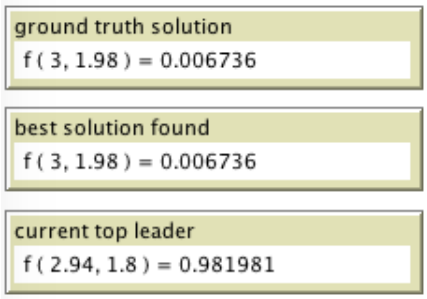


- SETUP: Computes the problem LANDSCAPE and displays it in the view grid as a 2D discretised surface plot. In addition, it creates the swarm, scatters the agents randomly within the boundaries of the landscape, and initialises the algorithm global variables (depending on the GRID-SIZE this setup may take a while).
- RESET: Relocates the agents randomly in the 2D view and initialises the algorithm's global variables (this is much faster than SETUP, as it doesn't recalculate the problem landscape, which might be useful for multiple simulation runs).
- GO: Iteratively execute the main procedure of the algorithm until finding the optimum or reaching the MAX-TICKS.
- STEP: Execute an iteration (a single tick) of the main procedure of the algorithm.
- SPOTLIGHT: A useful chooser to keep track of the optimal solution, the best solution found so far, or the current top leader within the display area.
- Also remember that the SPEED slider at the top of the window can be used to control how fast the simulation runs in the view grid.

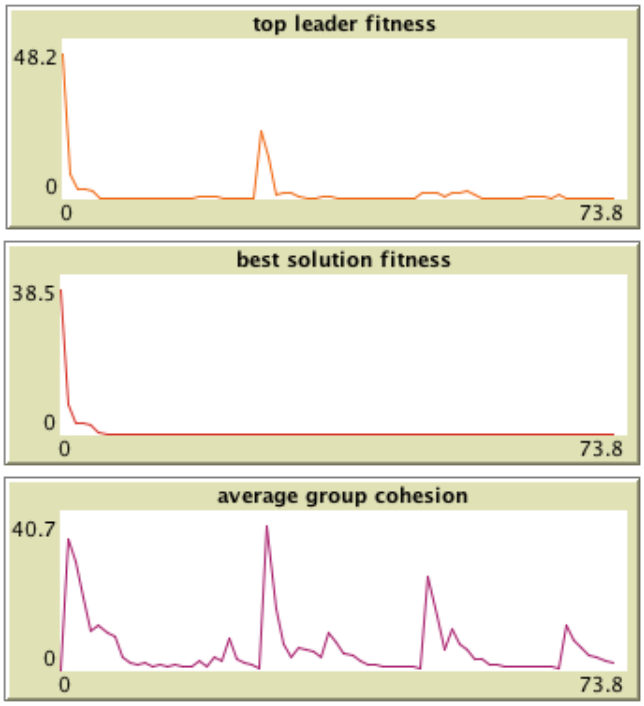
Once you start running the simulation, the swarm behaviour will emerge in the view grid (see below), as the agents try to discover the promising regions within the landscape and hopefully find the optimal patch before MAX-TICKS.



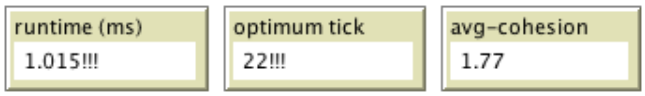
Besides, the output monitors (shown below) allow you to inspect such emerging behaviour as the model evolves:



- GROUND-TRUTH-SOLUTION: Coordinates and cost function value of the optimum.
- BEST-SOLUTION-FOUND: Coordinates and cost function value of the FLW-BEST-PATCH discovered so far.
- CURRENT-TOP-LEADER: Coordinates and cost function value of the leader at the best patch in the current iteration.



- TOP-LEADER-FITNESS: Historical plot of the leader with best fitness (value of cost function) throughout the simulation.
- BEST-SOLUTION-FITNESS: Historical plot of best solutions discovered throughout the simulation.
- AVERAGE-GROUP-COHESION: Historical plot of the average cohesion measurement throughout the simulation (only if COHESION? is enabled).



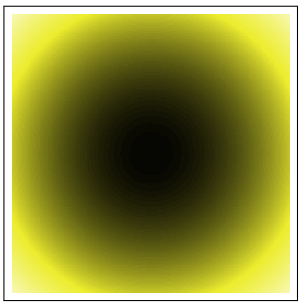
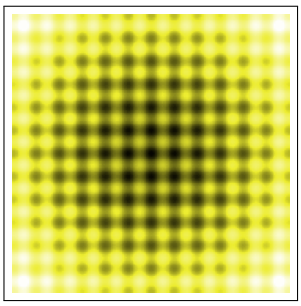
- RUNTIME: Total execution time of the simulation run.
- OPTIMUM-TICK: The tick number where the optimum was discovered (if it happens). Notice that if the algorithm is able to do so, these last two monitors will display a "!!!" sign in their text box.
- AVG-COHESION: The average measure of cohesion of the groups within the swarm (only if COHESION? is activated).

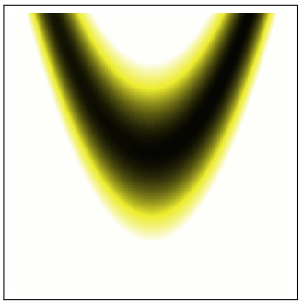
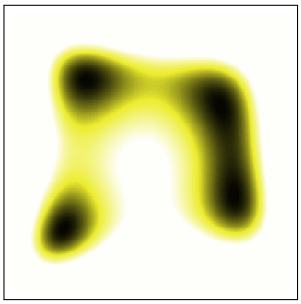
1.4 Other distinctive features

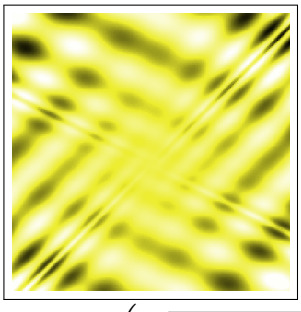
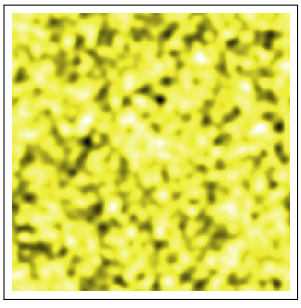
Each LANDSCAPE may exhibit different properties: multimodality, convexity, separability, multiple global or local minima, plateaus, ridges, valleys, etc. The advantage of swarm optimisation methods is their ability to explore several regions of the search space simultaneously so they can discover these properties without getting stuck in local minima, as opposed to single point metaheuristics. However, it is also known that these swarm-based methods may suffer from premature convergence where the entire population collapses to a single suboptimal patch or small region. For reference, each landscape exhibits the following properties (the actual mathematical expressions and 2D landscapes can be seen in the figure next page):

- SPHERE, SPHERE-OFFSET: unimodal, convex.
- RASTRIGIN: multimodal, multiple local minima, ridges.
- ROSENBROCK: multiple local minima, valleys.
- HIMMELBLAU: multimodal, multiple global minima, valleys.
- EGGHOLDER: multimodal, multiple local minima, valleys.
- RANDOM: multimodal, multiple local minima, sampled from a scaled normal distribution.

The FLW algorithm attempts to circumvent such swarm-based drawbacks by using two features. First, it allows for multiple leaders, meaning you can choose between having a single swarm of agents with an unique leader, or having multiple leaders with different subgroups of followers (distinguished by different colors). When choosing the multi-leader mode, you can notice how the subgroups spread out looking for minima in separate regions of the LANDSCAPE, which is partly due to the anti-clash mechanism that the algorithm implements to avoid collisions between leaders. You can test this emergent behaviour (leaders dominating separate regions of the search space) by experimenting with different values for the N-LEADERS and SWARM-SIZE sliders.

SPHERE Separable, Unimodal	RASTRIGIN Separable, Multimodal, Local minima
	
$f(\mathbf{x}) = x_1^2 + x_2^2$ $6 < x_1, x_2 < 6$ $\mathbf{x}^\star = (0,0);\ f(\mathbf{x}^\star) = 0$	$f(\mathbf{x}) = 20 + \sum_{i=1}^{2}(x_i^2 \quad 10cos(2\pi x_i))$ $6 < x_1, x_2 < 6$ $\mathbf{x}^\star = (0,0),\ f(\mathbf{x}^\star) = 0$

ROSENBROCK Non-Separable, Unimodal, Valley	HIMMELBLAU Non-Sep., Multimod., Non-local min.
	
$f(\mathbf{x}) = 100(x_2 \quad x_1^2)^2 + (1 \quad x_1)^2$ $6 < x_1, x_2 < 6$ $\mathbf{x}^\star = (1,1)$ $f(\mathbf{x}^\star) = 0$	$f(\mathbf{x}) = (x_1^2 + x_2 \quad 11)^2 + (x_1 + x_2^2 \quad 7)^2$ $6 < x_1, x_2 < 6$ $\mathbf{x}^\star = \begin{cases} (3,2), (2.81, 3.28) \\ (3.78, 3.28), (3.58, 1.85) \end{cases}$ $f(\mathbf{x}^\star) = 0$

EGGHOLDER Non-Separable, Multimodal, Local min.	RANDOM (example) Separable, Multimodal, Local min.
	
$f(\mathbf{x}) = \quad x_1 \sin\left(\sqrt{\lvert x_1 \quad (x_2 + 47)\rvert}\right)$ $(x_2 + 47)\sin\left(\sqrt{\lvert 0.5x_1 + (x_2 + 47)\rvert}\right)$ $512 < x_1, x_2 < 512$ $\mathbf{x}^\star = (512, 404.23),\ f(\mathbf{x}^\star) = \quad 959.64$	$f(\mathbf{x}) \quad \mathcal{N}((0,0), 500)$ $6 < x_1, x_2 < 6$ $\mathbf{x}^\star$: not preset (generated on-the-fly)

The second feature is intended to prevent premature convergence by dispersing agents by means of a warm restart of agent locations every time AVG-COHESION drops below a certain threshold (if COHESION? is enabled), or periodically once a predetermined number of ticks is reached (if COHESION? is disabled) during the execution of the GO loop. Since relocated leaders can start exploring newer unknown regions, an ELITISM-like learning mechanism allows the swarm to keep track of the most promising solution found so far in previous restarts by moving an arbitrary WALKER to the BEST-SOLUTION-FOUND patch; this way, eventually a leader would be attracted to that patch if no better solutions are found in the new restart.

You can test the above behaviour by experimenting with the COHESION? and ELITISM? switches, whose effect is depicted in the view area and in the periodic saw-like patterns that appear in AVERAGE-GROUP-COHESION and TOP-LEADER-FITNESS plots. Use the SPEED slider at the top to slow down the display of the warm restart mechanism. You can use the SPOTLIGHT chooser to highlight in the view area the location of the LANDSCAPE's GROUND-TRUTH-SOLUTION or the current BEST-SOLUTION-FOUND patch or the current TOP-LEADER.

1.5 Try it yourself

The following are interesting simulation scenarios:

- Solve each LANDSCAPE with a single-leader swarm. Repeat the experiment many times (use the SETUP button once, then RESET and GO as many times as you like). What is the success rate for each problem? (i.e. what percentage of the total number of repetitions was the FLW algorithm able to discover the ground-truth optimum?)
- Is the multi-leader strategy better than the single-leader strategy? Which one achieves a higher success rate? Do you get better results as the number of leaders increases? You can test the last hypothesis by trying different combinations of pair values (N-LEADERS, SWARM-SIZE): (1, 5), (2, 10), (4, 20), (8, 40).
- Try to solve each LANDSCAPE with different resolutions (GRID-SIZE): 100x100, 200x200, 400x400, 800x800. This parameter affects the sampling rate of the problem cost function during quantisation and thus changes the size of the search space (the larger this size, the more difficult the search for the best patch). How does this affect the success rate of the algorithm? How does this affect the speed of convergence (that is, the average tick where the optimum is discovered)?
- Notice that unlike the other problems, the RANDOM problem produces a different landscape each time you press SETUP. It would be interesting to investigate if nonetheless the FLW algorithm is able to solve it or if some instances present more difficulty than others.

As a side note, we remark that the resolution level can induce quantisation errors during the cost function sampling, therefore the optimum patch coordinates of a given LANDSCAPE can differ depending on the GRID-SIZE. For example, the optimum of ROSENBROCK'S problem for 100x100 and 200x200 resolutions is at *f(1.2, 1.44) = 0.04*, but for 400x400 resolution it is at *f(0.93, 0.87) = 0.007501*, whereas for 800x800 resolution it is at *f(1, 1) = 0.00255*.

Another interesting idea to experiment with is to assess the performance of the model under different swarm configurations, in terms of efficiency. Since the worst case for a single-agent algorithm would be to perform an exhaustive search of the entire LANDSCAPE, i.e., evaluate the cost function on the entire grid of NxN patches (where N depends on the GRID-SIZE), the efficiency of each algorithm configuration can be measured as the percentage of evaluations of the cost function needed to find the optimum, with respect to NxN, in other words, the average rate OPTIMUM-TICK / NxN. The following swarm configurations can be tested:

- *Simultaneous Global Random Search*: A swarm of WALKERs (SWARM-SIZE: 5, N-LEADERS: 1, WALKERS-RATE: 0.8, COHESION?: OFF, ELITISM?: OFF)
- *Simultaneous Local Random Search*: A swarm of singleton LEADERs (SWARM-SIZE: 5, N-LEADERS: 5, WALKERS-RATE: 0.2, COHESION?: OFF, ELITISM?: OFF)
- *Single-leader Swarm Search*: A swarm of agents with an unique LEADER, FOLLOWERs and WALKERs (SWARM-SIZE: 5, N-LEADERS: 1, WALKERS-RATE: 0.2, COHESION?: OFF, ELITISM?: OFF).
- *Multi-leader Swarm Search*: A swarm of agents with multiple LEADERs, FOLLOWERs and WALKERs (SWARM-SIZE: 20, N-LEADERS: 4, WALKERS-RATE: 0.2, COHESION?: OFF, ELITISM?: OFF)
- *FLW full swarm search*: A swarm of agents with multiple LEADERs, FOLLOWERs and WALKERs, plus cohesion tracking and elitism (SWARM-SIZE: 20, N-LEADERS: 4, WALKER-RATE: 0.2, COHESION?: ON, ELITISM?: ON)

Can you identify which configuration (hence which search strategy/operators) improves the optimisation effectiveness/efficiency of the FLW algorithm? Is there a minimum follower base size for the multi-leader setup to work properly?

1.6 Extending the tool

Some possible paths for tool extensions are:

- Expand the landscape repertoire with additional problem cost functions.
- Extend the model to solve problems in higher dimensions (not necessarily using a 2D grid view).
- Extend the model to discrete (binary) problem domains.
- Extend the model to solve non-stationary problems, that is, problems where the landscape can vary with time during the course of a run. In this sense, we believe that the mechanisms to prevent premature convergence implemented in the FLW algorithm can be useful to also adapt to this type of dynamic changes in the landscape.

Chapter 2

Installation and execution

2.1 Online version

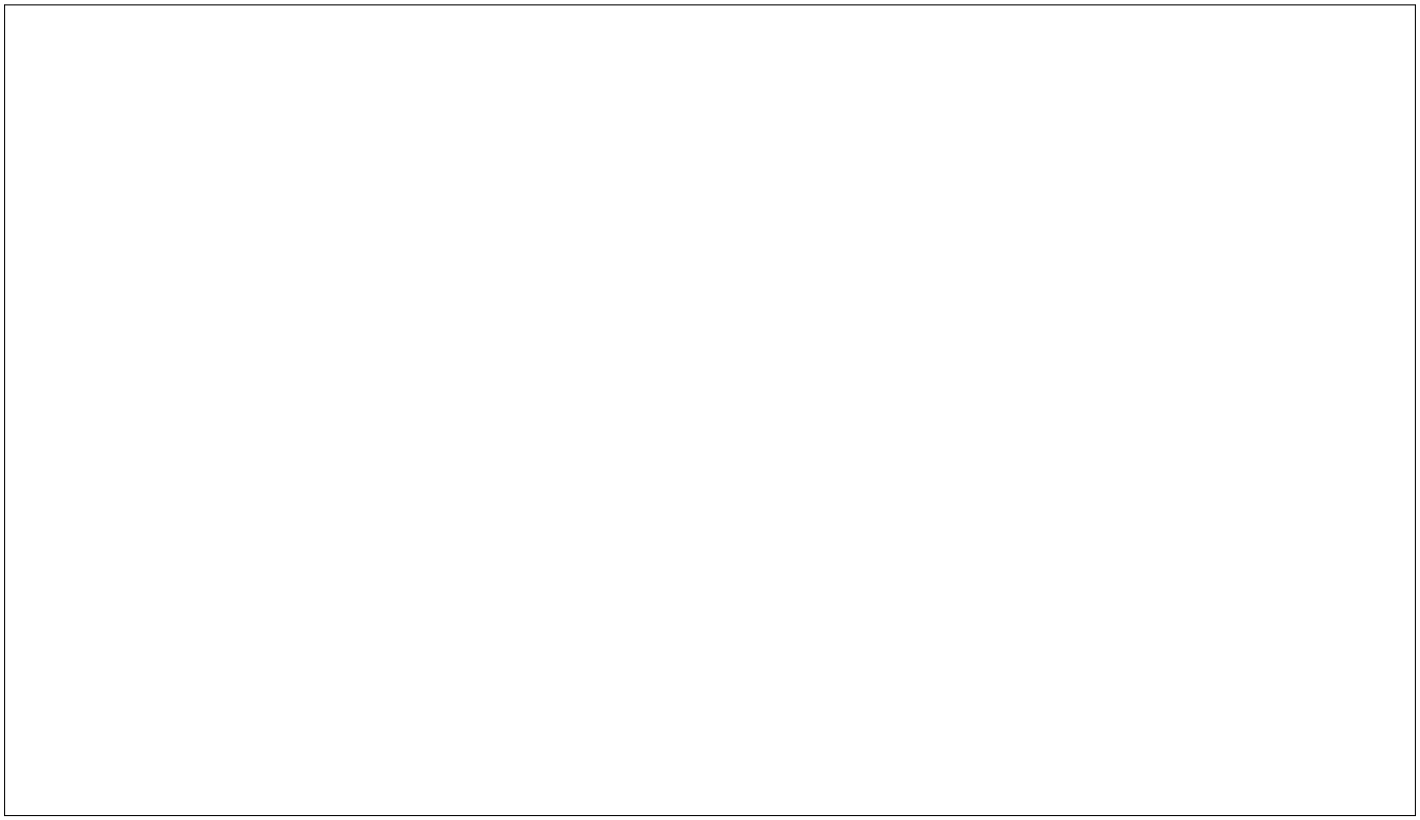
The easiest way of experimenting with FLAMEGPU Swarm is by using its online version. The software is available at the ModellingCommons website. So, you just need to follow these steps:

1. Open your favourite Internet browser and point it to the following URL: http://modelingcommons.org/browse/one_model/6978
2. The following web page should appear:

3. From the toolbar, choose the “Run in Netlogo Web” tab:

4. A grey area in the middle of the screen is shown. Do “Click to Run Model”:

5. The model main screen will show up:



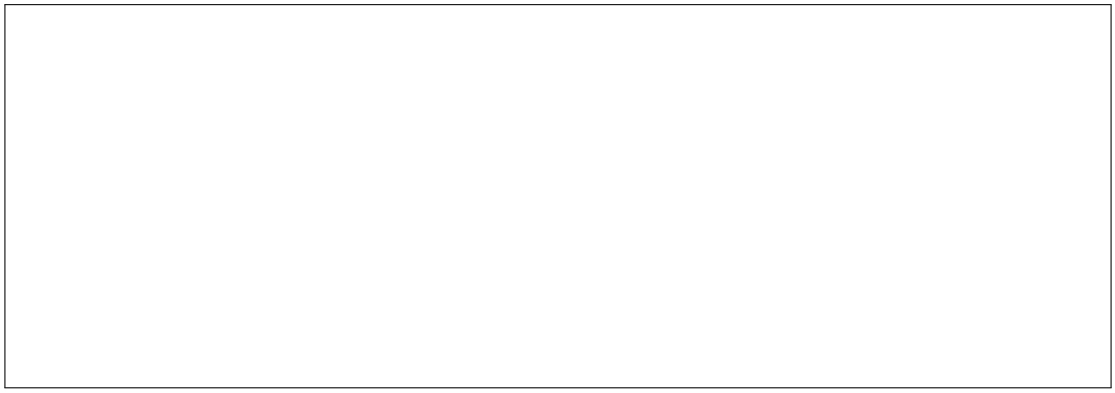
6. That's all! Choose the running parameters in the control panel, click SETUP and then GO! You will see how the swarm of agents adapt to the LANDSCAPE of the problem in the simulation view area, while the performance indicators will be shown in the monitors and the plots.

2.2 Desktop version

The desktop version is recommended if you want to try heavy experimentation, such as parameter tuning, average behaviour of multiple runs or simulations with large resolution for the view area. For this purpose, FLWSwarm runs over the NetLogo desktop simulation platform. In this case, you need to go through the following steps:

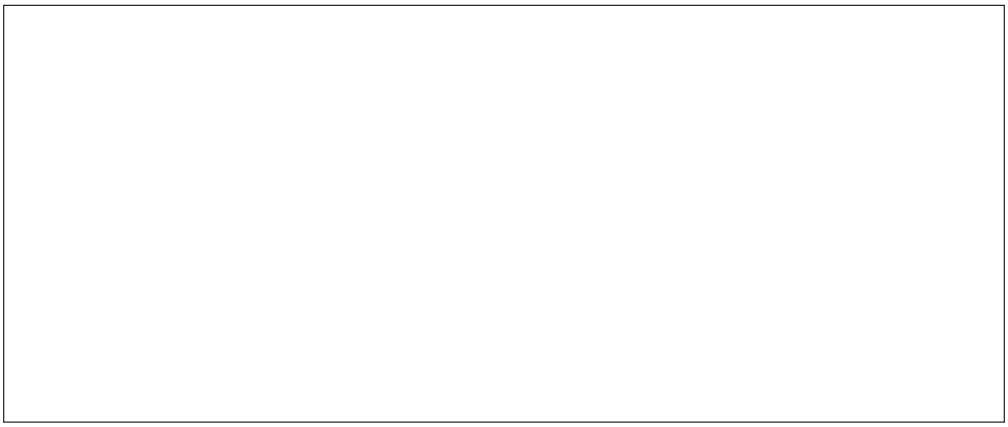
1. Download and install the NetLogo desktop software. For this purpose, go to http://ccl.northwestern.edu/netlogo/ , click in "Download NetLogo" and follow the installation instructions:

2. Download the FLW Swarm model file from the model webpage, using the "Export" button:



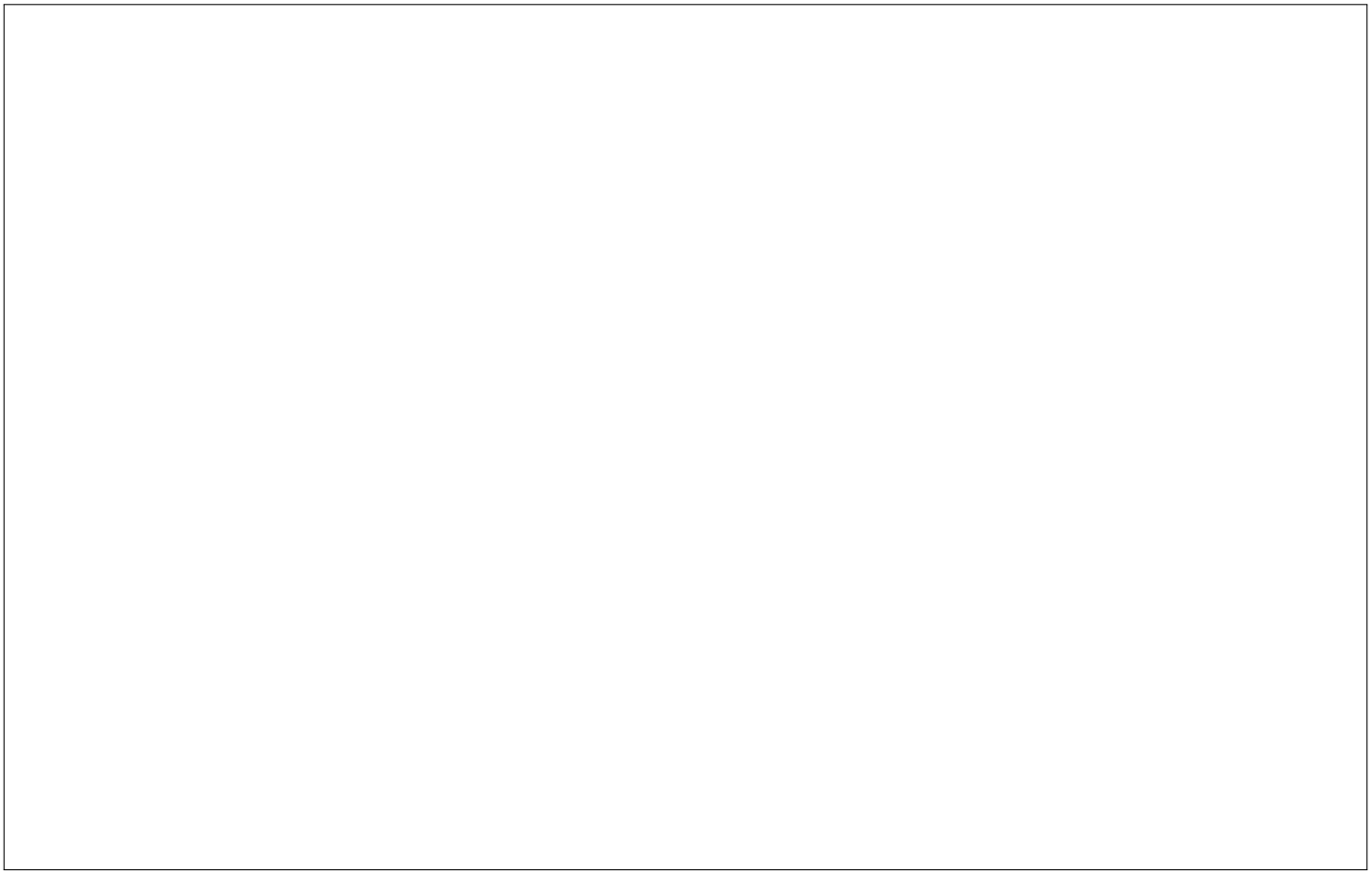
A file called Follow the Leader Random Walk (FLW) Swarm Algorithm.nlogo would be downloaded to your local drive.

3. Run NetLogo on your computer. Choose the menu option File! Open



Locate the Follow the Leader Random Walk (FLW) Swarm Algorithm.nlogo that you downloaded previously and open it.

4. The FLW Swarm desktop screen will show up:



That's it! Choose the running parameters in the control panel, click SETUP and then GO! You will see the swarm of agents adapting to the LANDSCAPE of the problem in the simulation view area, while the performance indicators will be shown in the monitors and the plots.