Wolf Pack Search Algorithm

Introduction

Global optimization is a necessary tool to have with wide range of applications in many areas, like in science, economy, engineering etc. But an increase in complexity of many real-world problems over time, and even the complexity of problems today, makes global optimization a challenging task, especially using traditional methods. In search of new ways and insights for tackling these issues, researchers have set themselves in search of inspiration from nature. There are many swarm intelligence phenomenon in nature, and scientist and naturalists got their attention piqued by the remarkable social and swarm behavior of animals such as swarming ants, schooling fish, and flocking birds.

One such marvelous swarm behavior is shown by a wolf pack. Harsh living environment and constant evolution for centuries have created their rigorous organization system and subtle hunting behavior. Attempts at mimicking the wolves tactics were made as early as back in Genghis Khan period. In 2011, a formal Wolf Colony Algorithm (WCA) was proposed, which fell short due to its low accuracy and efficiency, and easily falling into local minima. In 2014, an improved algorithm was proposed under the name Wolf Pack Algorithm (WPA).

Analyzing of Wolf Pack

A wolf pack shows an impressive implementation of both social hierarchy and ferocious hunting mechanism. The social structure of a wolf pack shows us clear social work division. Firstly, there is a lead wolf, the smartest and most ferocious in the pack, selected by natural selection, a leader under the law of the jungle. Lead wolf commands other wolves in the pack, takes decisions for the pack by evaluating surrounding situations, and is responsible for ensuring the pack’s survival.

Secondly, there are some elite wolves, who obey the lead wolf but have a higher social status and responsibilities than the rest of the pack. They are called scouts because the lead wolf sends them to look for prey in a probable scope. They wander around following the smell left by the prey. This is referred to as Scouting.

Once a scout wolf finds trace of prey, it howls and informs the lead wolf, who will evaluate the situations and decide whether to pursue the trace or not. If deciding to follow the prey, the lead wolf will howl and summon the rest of the pack, the remaining ferocious wolves in the pack. The ferocious wolves will move swiftly towards the location of the lead wolf (by following the howling) while the lead wolf will be periodically howling and moving towards the scout wolf closest to the prey. This is referred to as Calling.

After running towards the prey with big strides, all the wolves except the lead wolf stop at a certain distance away from the prey, and assume a Besieging position. The lead wolf then goes in alone and captures the prey.

The captured prey is distributed from the strongest to the weakest. This will result in death of some weak wolves, but ensures that the strongest wolves survive and grow strong to benefit the pack in the next hunt. The dead members of the pack are later replaced by new recruits.

Algorithm

WPA has the benefit of using three artificial intelligence behaviors (Scouting, Calling, Besieging) and two intelligence rules (winner-take-all rule for generating lead wolf, and the stronger-survive renewing rule for wolf pack). The scouting behavior accelerates the possibility that WPA can fully traverse the solution space. Once the winner-take-all rule generates a lead wolf, the Calling behavior makes the wolves move towards the lead wolf whose position is nearest to the optimal solution.

Measures taken for performance boosting of the technique brings in a few differences in the actual algorithm used in comparison to what we see in nature. We employ all wolves in the Scouting process to find the optimal solution faster. Instead of sticking to one specific lead wolf, we test each wolf for fitness in each iteration, and if a wolf other than the previously assigned lead wolf has a better fitness value, we assign this wolf as the new lead wolf. The typical WPA algorithm is described below in computation steps.

Step 1 (Initialization): Initialize the necessary parameters, like number of wolves, positions of wolves, maximum number of iterations, maximum number of scouting repetitions, step coefficient, distance determinant coefficient, population renewing proportional coefficient etc.

Step 2 (Scouting): After choosing the fittest wolf to be the lead wolf, the rest of the wolves take scouting behavior and search the solution space, until either one of the wolves find a better solution than the lead wolf, or until maximum number of repetitions is reached.

Step 3 (Calling): The wolf that is closer to the optimal solution is now assigned the new lead wolf, who calls the rest of the pack towards its position. All the wolves gather towards the lead wolf fast, until either a different wolf finds a better solution on the way (in which case that wolf takes the lead position and Calling behavior), or until they are within a specific distance from the lead wolf.

Step 4 (Besieging): All the wolves who reach the specific distance from the lead wolf now stop and take Besieging behavior.

Step 5 (Renew the position of lead wolf): Lead wolf finds the optimal solution in this localized space that contains the global optimum, and the position of lead wolf is updated according to winner-take-all rule.

Step 6 (Renew wolf pack): Wolf pack is updated under the population renewing rule.

Step 7 (Termination): If the solution matches the precision requirement or if the maximum number of iterations is reached, the program terminates and outputs the optimal solution. Else the program goes back to Scouting (step 2).

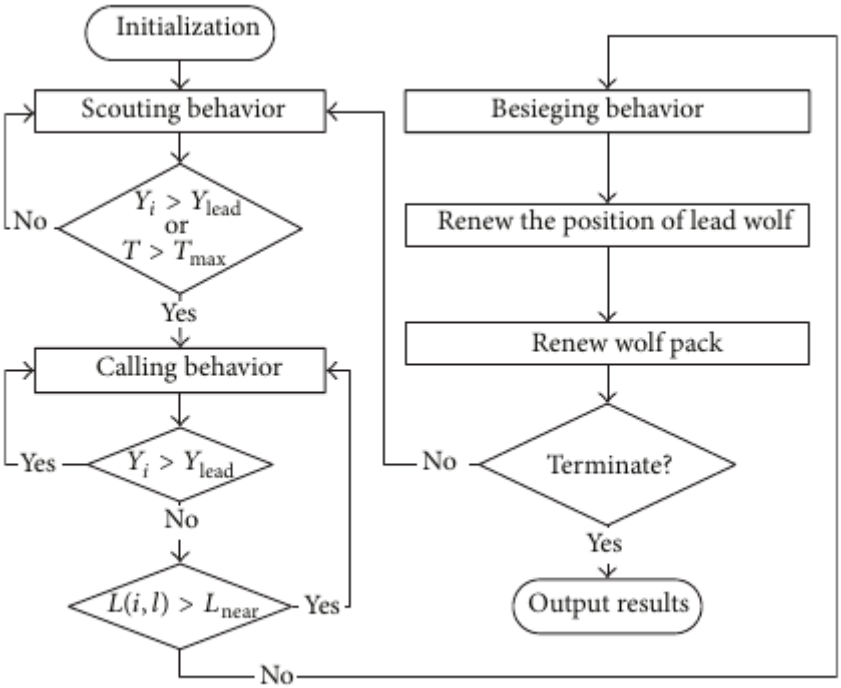


FIGURE 1: The flow chart of WPA.

The flow chart of WPA is shown above.

Improvement

In search of improving the efficiency of WPA, changes were proposed. The wolf tactic variation Grey Wolf Optimization (GWO) algorithm brings in interesting performance boosts regarding the run-time. While in WPA the three artificial intelligence behaviors are executed sequentially in different iteration loops, GWO executes all three behaviors sequentially once in each iteration loop. Moreover, instead of following one lead wolf, GWO uses three fittest wolves (alpha, beta and delta) with appropriate weight biases to control the wolf pack behaviors, with alpha being the lead wolf. This is done by controlling the movement of the alpha, beta, delta wolves with a parameter A (randomized) and then the movement of the pack with respect to alpha, beta, delta wolves with a parameter C (randomized), and then averaging the positions.

Instead of using GWO, we have started with writing WPA and worked to improve the efficiency of WPA by referring to GWO and by reducing run-time by optimizing the flow of control of the code.

Implementation

The algorithm was written in Python programming language. The program can either be run with default parameters using “python WolfpackAlgorithm.py”, or be imported as a module. The module (documented with “help”) consists of a class “wpa” which can take as arguments the number of wolves, function to be optimized, lower limits of plot axis, upper limits of plot axis, dimensions and maximum number of allowed iterations. The program run will return an object which fetches positions of each wolves in each iteration (wpa.get_wolves()) and optimal solution position (wpa.get_Lead()). Few benchmark functions are imported from “testFunctions.py” (attached).

Benchmark problems and solutions

The algorithm is tested against some benchmark functions (listed below) of known optimal solutions for measuring the accuracy.

Functions	Formulation	Global extremum	D
Rosenbrock	$f(\vec{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$f_{\min}(\vec{x}) = 0$	2
Sphere	$f(\vec{x}) = \sum_{i=1}^{} x_i^2$	$f_{\min}(\vec{x}) = 0$	200
Sumsquares	$f(\vec{x}) = \sum_{i=1}^{D} i x_i^2$	$f_{\min}(\vec{x}) = 0$	150
Booth	$f(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$f_{\min}(\vec{x}) = 0$	2
Ackley	$f(\vec{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i\right) + 20 + e$	$f_{\min}(\vec{x}) = 0$	50

Table of Benchmark functions

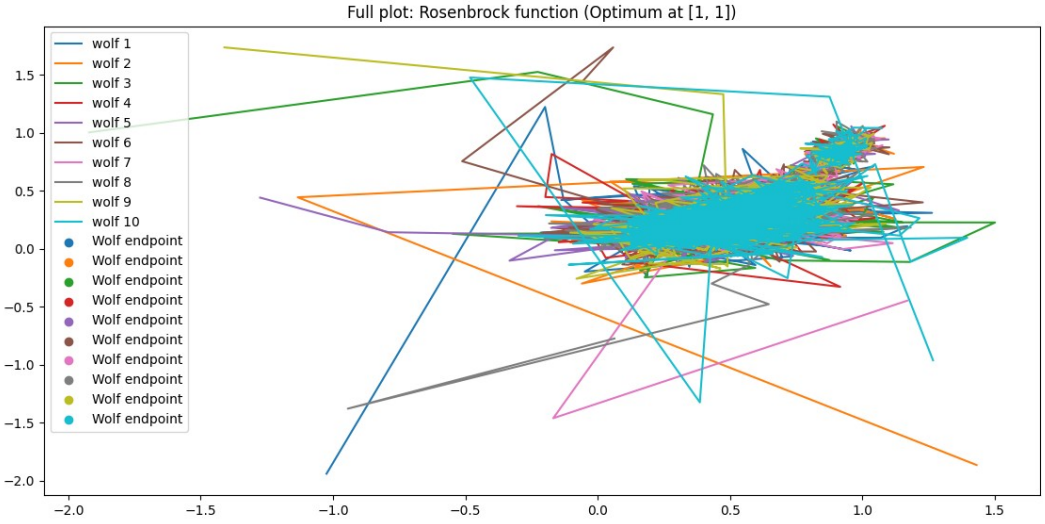
Following are the results of the algorithm optimizing the above benchmark functions. Positions of global extremum and computed extremum are tabulated, with the error in the position computed.

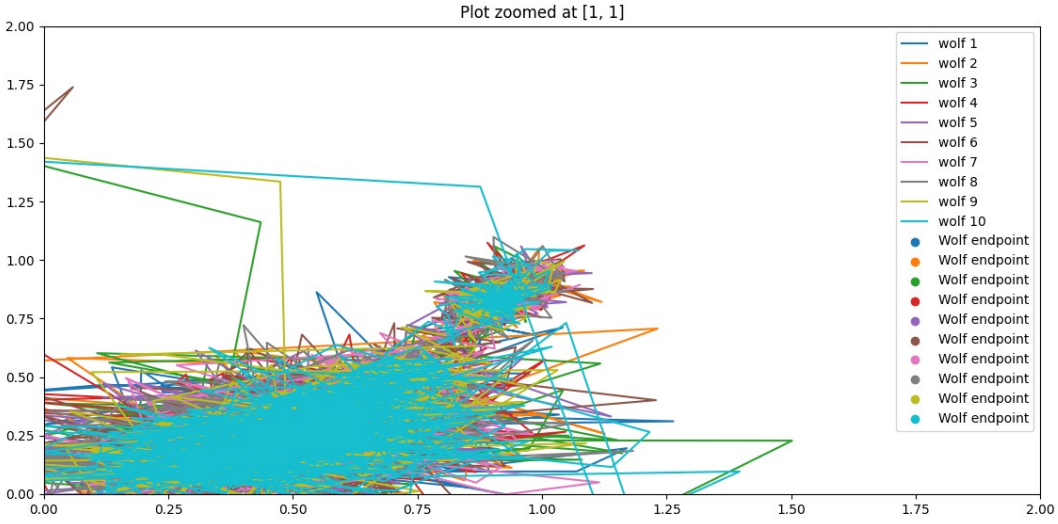
Benchmark function	**Global extremum**	**Computed extremum**	**Error**
Rosenbrock	[1, 1]	[0.9896482892484185, 0.9801718435831055]	2.2367693273496983e-2
Sphere	[0, 0]	[-7.167324431374554e-112, 7.659159885262394e-112]	1.0489674430247644e-111
Sumsquare	[0, 0]	[-1.3008303238511788e-120, -9.379126664194878e-121]	1.6036956386050553e-120
Booth	[1, 3]	[1.0004415836667067, 2.999497689530941]	6.6881383211494450e-4
Ackley	[0, 0]	[1.561913353046971e-17, -2.0143480486045753e-16]	2.0203944649848268e-16

The points of extremum are computed for each of the above benchmark functions, and found to be of minimal error.

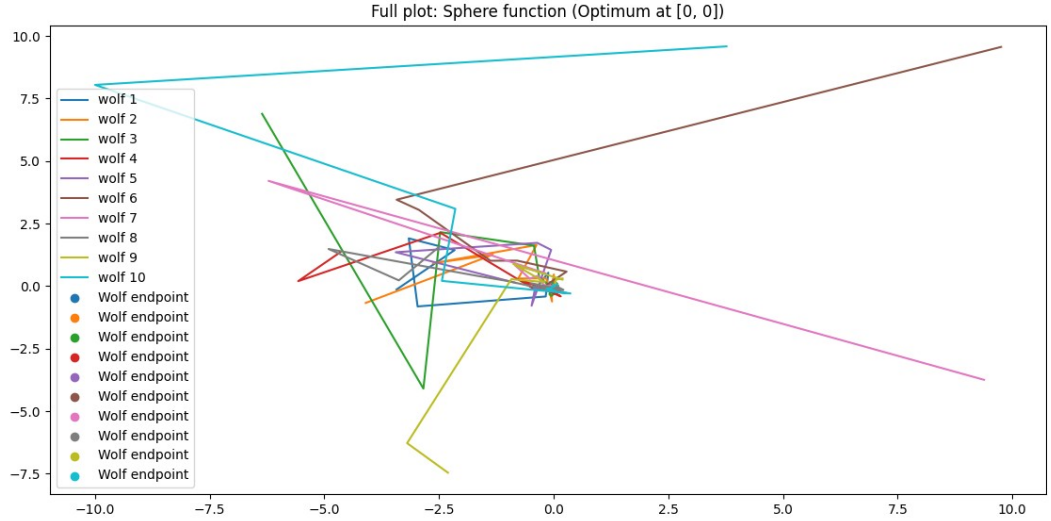
Following are the graphical 2D plot for the above benchmark functions solved optimized using WPA.

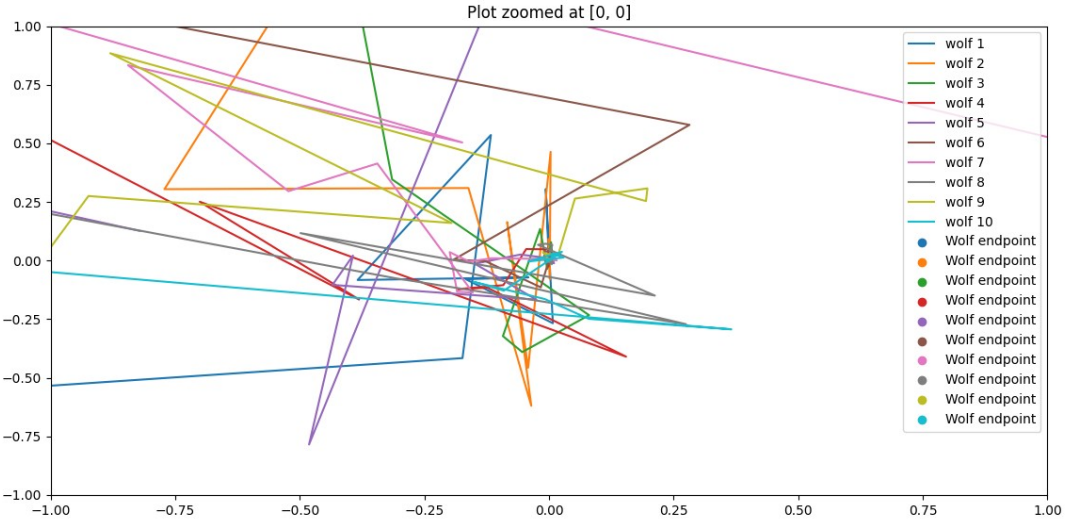
Rosenbrock function



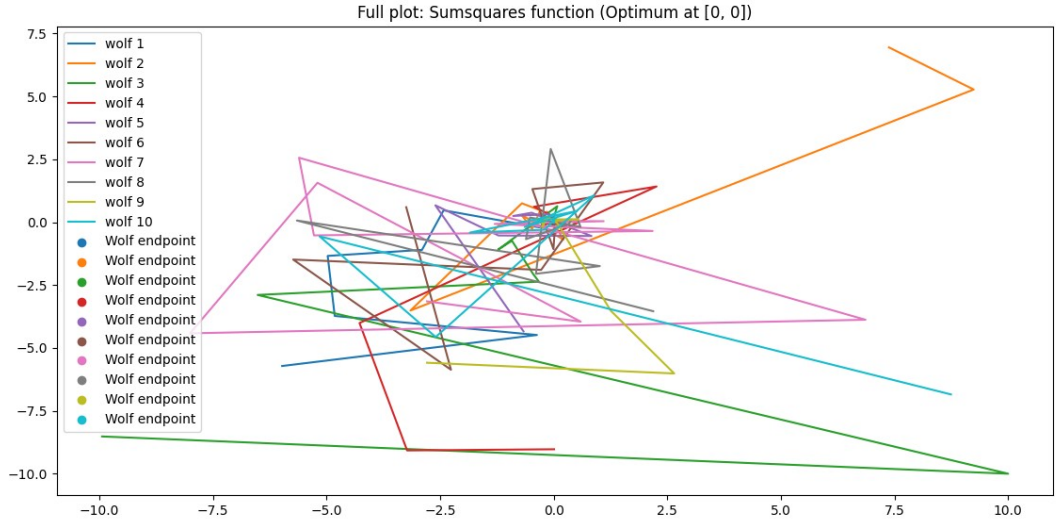


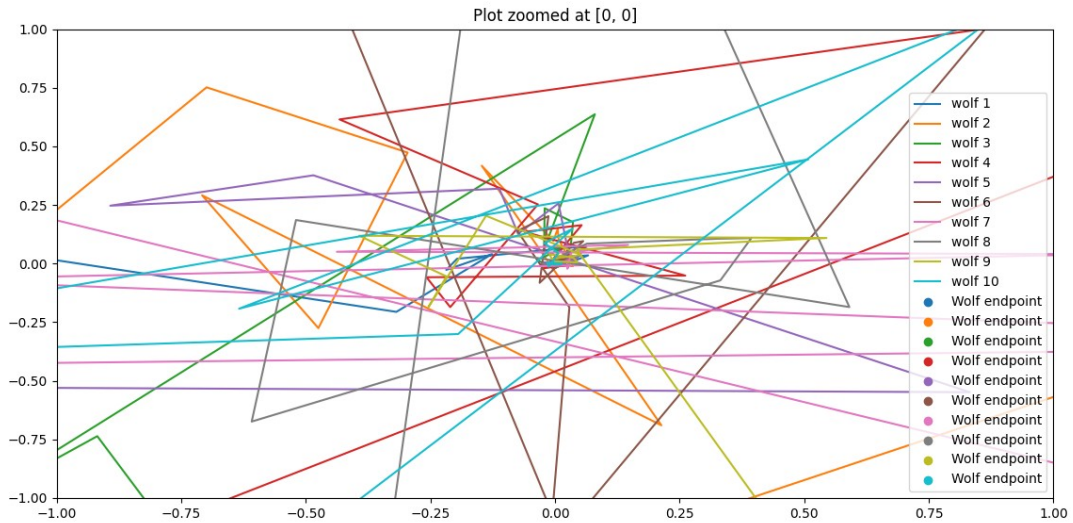
Sphere function



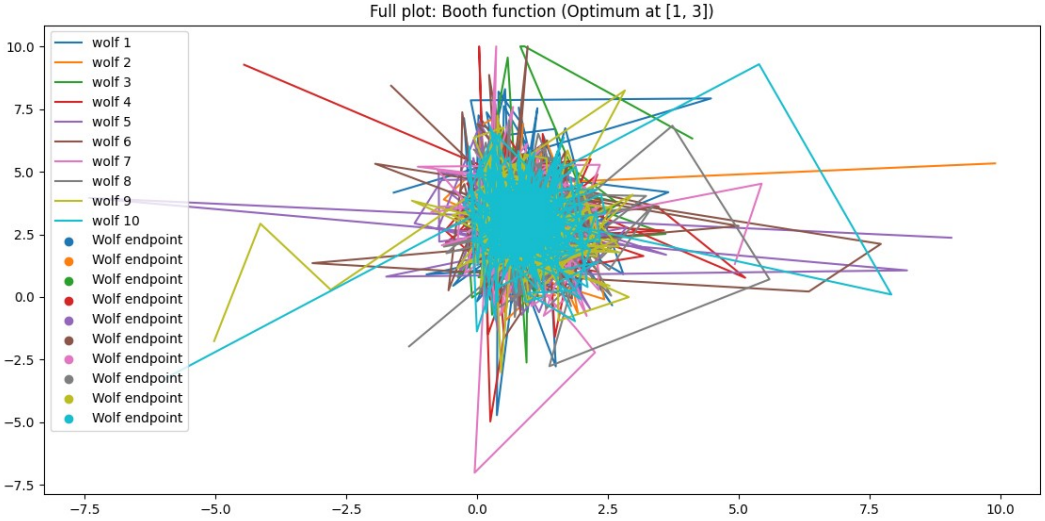


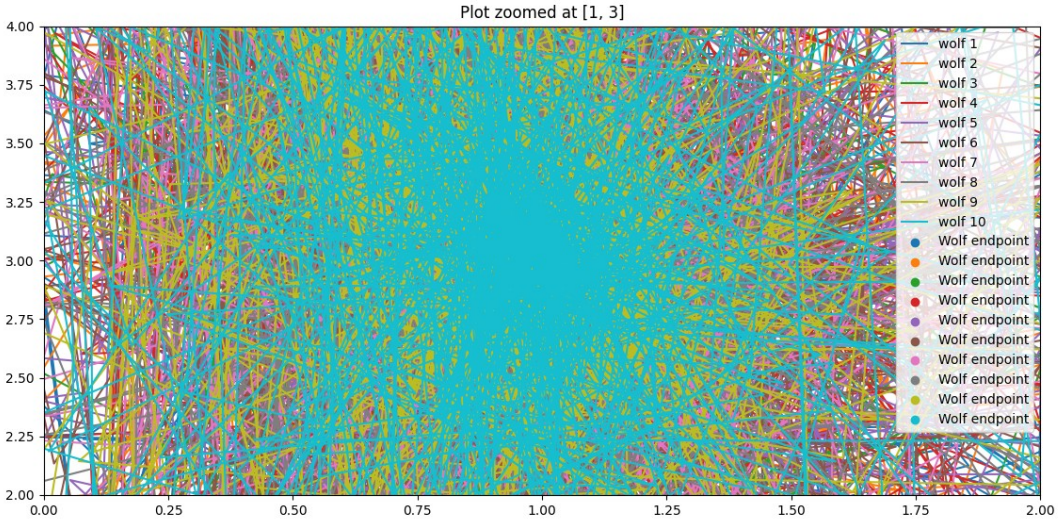
Sumsquares function



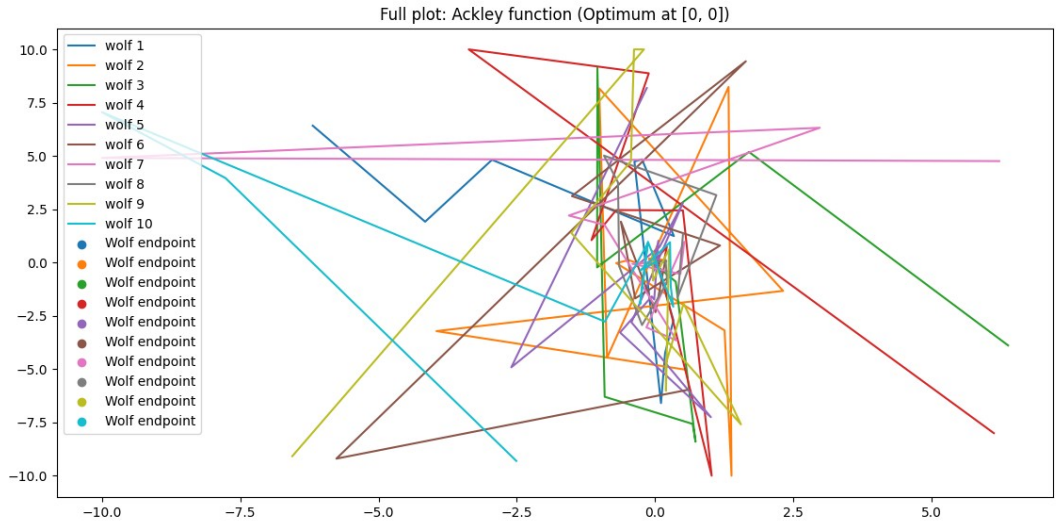


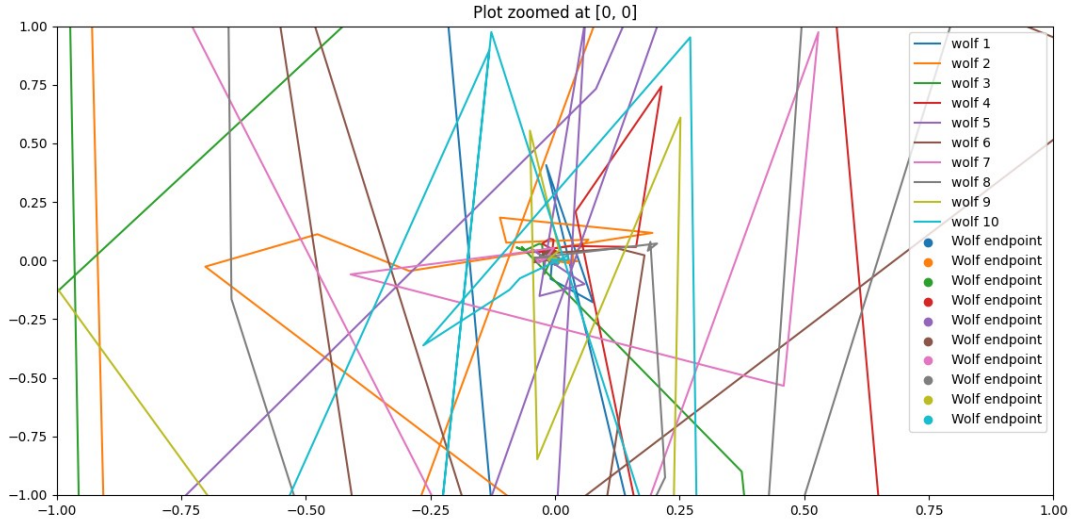
Booth function





Ackley function





Conclusions

The basics and principles of Wolf Pack search Algorithm (WPA) were studied. The algorithm was coded in Python, with improvements made on performance. The program was tested with few benchmark functions, and the results were tabulated. The program seems to deliver results of dependable accuracy. This swarm intelligence method provides a very good convergence performance.

References

- Yang et. al., “Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search”, *International Conference on Intelligent Pervasive Computing*, 2007.
- H. Wu and F. Zhang, “Wolf Pack Algorithm for Unconstrained Global Optimization”, *Mathematical Problems in Engineering*, vol. 2014, Article ID 465082, 17 pages, 2014.