Author: Jacob Olinger
Email: jacoboli@pdx.edu
Project Name: Rust-Lion
Repo: https://github.com/rhit-olingejj/rust-lion

The proposed project focuses on implementing the Lion Algorithm, a bio-inspired swarm optimization technique, in Rust. The Lion Algorithm is modeled on the social organization and hunting behavior of lions, where populations are divided into prides and nomads that cooperate and compete for dominance. Lions' natural processes, such as territorial defense, mating, and cub rearing, are abstracted into computational operators that balance exploration and exploitation in search spaces. I became interested in nature-inspired optimization methods after studying the firefly algorithm a few summers ago at NDSU. The Lion Algorithm has been applied to a range of continuous and combinatorial optimization problems and represents an excellent candidate for demonstrating Rust's strengths in safety, performance, and concurrency. Alongside the algorithm's core implementation, the project will include basic visualizations to help users observe the population's evolution, territorial shifts, and convergence patterns across various datasets.

The vision for this project is to create a modular, efficient, and extensible Rust library for the Lion Algorithm that supports experimentation and visualization. The project will produce two main deliverables: a core Rust crate implementing the Lion Algorithm with configurable parameters, and a visualization component for tracking population behavior and performance over time. The visualization will show the distribution of lion prides and nomads as they search for optimal solutions. The system will also generate performance metrics to allow users to evaluate the algorithm's stability and efficiency under different configurations. To support flexibility in experimentation, the implementation will accept data input from either randomly generated parameters or external data files.

The core implementation will capture the Lion Algorithm's distinctive structure, which divides the population into prides and nomads. Lions in prides undergo cooperative behaviors such as mating and defending territory, while nomadic lions explore new regions of the search space, providing global search capability. The algorithm's operators, territorial takeover, mating, roaming, and cub growth, will be implemented as modular functions. Key parameters such as pride ratio, nomad ratio, mating rate, and territorial takeover probability will be adjustable via command-line arguments. Rust's concurrency features will be leveraged to evaluate multiple lions or prides in parallel, improving performance for computationally intensive objective functions.

The visualization component will illustrate how the lion population evolves. Using contour plots, the algorithm will display prides as clusters and nomads as scattered agents exploring the search space. Users will be able to view how lions move, merge, or compete for territories as the

algorithm converges toward optimal solutions. The visualization system aims to make the algorithm's dynamics intuitive to follow, helping users understand how exploration and exploitation are balanced over iterations.

Several issues of concern need to be addressed in this project. First, the Lion Algorithm's performance can be highly sensitive to parameter settings, and inappropriate configurations can cause premature convergence or excessive exploration. To mitigate this, the project will include a set of default parameters and adaptive schedules that modify key values dynamically. Second, scaling the algorithm to high-dimensional problems can challenge performance, as the pride-nomad structure may become less efficient in large spaces due to the curse of dimensionality. The implementation will therefore target medium-dimensional problems, 10–100 dimensions, and may explore hybridization with local search strategies for higher dimensions. Third, visualization performance can be limited when rendering many agents in real time; this will be addressed by updating the display periodically rather than at every iteration and by offering an offline plotting option.

In summary, Rust-Lion aims to deliver a fast, extensible, and visual implementation of the Lion Algorithm in Rust, suitable for experimentation. Rust-Lion aims to contribute a robust and user-friendly framework to the Rust ecosystem, supporting future work in bio-inspired and population-based optimization algorithms.