Swarm and Evolutionary Based Algorithms used for Optimization

Augusto Mathias Adams*, Caio Phillipe Mizerkowski†, Christian Piltz Araújo‡ and Vinicius Eduardo dos Reis§

*GRR20172143, augusto.adams@ufpr.br, †GRR20166403, caiomizerkowski@gmail.com,
‡GRR20172197, christian0294@yahoo.com.br, §GRR20175957, eduardo.reis02@gmail.com

***Abstract*—In this paper, a study of evolution and swarm based algorithms is presented, using two classical engineering problems: *Spring Tension* and *Pressure Vessel Designs*. The test code for the problems was made using the *Python Language*, version 3.10 and uses *MealPy* package, version 2.5.1, to provide the algorithms. Thre algorithms were randomly chosen from a vaste list from MealPy algorithms: *Evolutionary Programming (LevyEP)*, *Evolution Strategies (OriginalES)* and *Genetic Algorithm (BaseGA)* from *evolutionary_based* subpackage; *Bees Algorithm (OriginalBeesA)*, *Firefly Algorithm (OriginalFFA)* and *Particle Swarm Optimization (OriginalPSO)* from *swarm_based* subpackage. Each problem was modeled using *SciPy* package, with constraints implemented as *penalty functions*. Each algorithm were optimized separately to extract the best solutions from each problem using the *MealPy*'s *Tuner* utility. The results, however, are dependant of algorithm and problem presented but from similarity test using Friedman's Chi-Squared test says that using any of the tested algorithm will produce similar results in the two optimization problems.**

***Index Terms*—Optimization Methods, Evolutionary Programming, Evolutionary and Swarm Based Strategies.**

I. Introduction

The study comprizes two great areas of artificial intelligence: Evolutionary and Swarm Inteligence algorithms.

Evolution: From Jason Brownlee's *"Clever Algorithms" - Evolutionary Algorithms belong to the Evolutionary Computation field of study concerned with computational methods inspired by the process and mechanisms of biological evolution. The process of evolution by means of natural selection (descent with modification) was proposed by Darwin to account for the variety of life and its suitability (adaptive fit) for its environment. The mechanisms of evolution describe how evolution actually takes place through the modification and propagation of genetic material (proteins). Evolutionary Algorithms are concerned with investigating computational systems that resemble simplified ver- sions of the processes and mechanisms of evolution toward achieving the effects of these processes and mechanisms, namely the development of adaptive systems. Additional subject areas that fall within the realm of Evolutionary Computation are algorithms that seek to exploit the properties from the related fields of Population Genetics, Population Ecology, Coevolutionary Biology, and Developmental Biology.*

Swarm Optimization: From Jason Brownlee's *"Clever Algorithms" - Swarm intelligence is the study of computational systems inspired by the 'collective intelligence'. Collective Intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. Examples include schools of fish, flocks of birds, and colonies of ants. Such intelligence is decentralized, self-organizing and distributed through out an environment. In nature such systems are commonly used to solve problems such as effective foraging for food, prey evading, or colony re-location. The information is typically stored throughout the participating homogeneous agents, or is stored or communicated in the environment itself such as through the use of pheromones in ants, dancing in bees, and proximity in fish and birds.*

II. Spring Tension Design

A. Evolutionary Algorithms

B. Swarm Based Algorithms

III. Pressure Vessel Design

A. Evolutionary Algorithms

B. Swarm Based Algorithms

IV. Discussion

The main objective of this work is demonstrate the inner work of a optimization class of algorithms called *Quasi-Newton algorithms*. Four of them are presented, along with the results and analysis regarding convergence, function evaluations and function objective values.

For implementation, it was decided to put constraints on search space, because for the original search space of some functions, like Ackley and Rastrigin, there are other minimas that will confuse every algorithm tested for this paper, and there is no sense on test a local minima procedure on these conditions. To restrain the initial solution to some region in the search space was a cheap and ready solution to avoid the algorithms to stop because of stumbling upon the wrong minimum.

The *Levenberg-Marquardt Algorithm* implemented for this paper is very sensitive to hyperparameters, namely α and λ, which was not tunned to every function, so, the conspicuous performance variation is noticeable along the results. LMA is known to be fast and reliable, but in our implementation it was found a little tricky to get the hyperparameters right for all the test functions. But for Rosenbrock and Rastrigin, it is the best option, even for lame implementations like ours.

The *Broyden–Fletcher–Goldfarb–Shanno* Algorithm was the only one that we decided to use it as is from *SciPy*, because

it has features desired for this work, like counting of function evaluations, gradient evaluation and iterations. It performed very well for Ackley and Booth Functions, and reasonably well for other functions.

The *Davidon–Fletcher–Powell* Algorithm is not implemented in *SciPy* but its equations shows that it is the dual of BFGS, so, it was easy to implement using some *SciPy* code for BFGS algorithm. In the DFP procedure, there is a Hessian inversion to be evaluated every iteration, then, numerical imprecision would be taken into consideration evaluating its results. While in some cases it has a superb performance (Like Ackley Function), it has very poor performance in Rosenbrock Function and even in Beale Function. The increase of dimensions makes DFP suffer of numerical imprecision due to the Hessian inversion, except for Rastrigin, and it is the loser algorithm in other functions.

Finally, the *Limited memory BFGS* has an average performance among the functions. The original algorithm was implemented using code found in the url https://github.com/qkolj/L-BFGS/blob/master/L-BFGS.ipynb.

In fact, there are no Quasi-Newton algorithms that solver the minima problem for an any function. From the standpoint of performance and convergence, all the algorithms have an average performance and a reasonable convergence, although we have to restrain the search space to a small area near the optimal solution.