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, it would be suffice to say that using any of the tested algorithm will produce similar results in these two optimization problems.**

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

I. Definitions

The main objective of this paper is study evolutionary and swarm intelligence algorithms. We present the main concepts of these two algorithm's classes, along with the chosen algorithms definitions in this section. All citations made in this document are due to Swarm Intelligence classes and to *MealPy*'s documentation, which points out the theoretical documentation for each implemented algorithm.

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 Intelligence: 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.

Evolutionary Programming: From Jason Brownlee's *"Clever Algorithms"* - Evolutionary Programming is a Global Optimization algorithm and is an instance of an Evolutionary Algorithm from the field of Evolutionary Computation. The approach is a sibling of other Evolutionary Algorithms such as the Genetic Algorithm, and Learning Classifier Systems. It is sometimes confused with Genetic Programming given the similarity in name, and more recently it shows a strong functional similarity to Evolution Strategies. Evolutionary Programming is inspired by the theory of evolution by means of natural selection. Specifically, the technique is inspired by macro-level or the species-level process of evolution (phenotype, hereditary, variation) and is not concerned with the genetic mechanisms of evolution (genome, chromosomes, genes, alleles).

Evolutionary Strategies: From Jason Brownlee's *"Clever Algorithms"* - Evolution Strategies is a global optimization algorithm and is an instance of an Evolutionary Algorithm from the field of Evolutionary Computation. Evolution Strategies is a sibling technique to other Evolutionary Algorithms such as Genetic Algorithms (Section 3.2), Genetic Programming (Section 3.3), Learning Classifier Systems, and Evolutionary Programming. A popular descendant of the Evolution Strategies algorithm is the Covariance Matrix Adaptation Evolution Strategies (CMA-ES).

Genetic Algorithms: From Jason Brownlee's *"Clever Algorithms"* - The Genetic Algorithm is an Adaptive Strategy and a Global Optimization technique. It is an Evolutionary Algorithm

and belongs to the broader study of Evolutionary Computation. The Genetic Algorithm is a sibling of other Evolutionary Algorithms such as Genetic Programming, Evolution Strategies, Evolutionary Programming, and Learning Classifier Systems. The Genetic Algorithm is a parent of a large number of variant techniques and sub-fields too numerous to list. The Genetic Algorithm is inspired by population genetics (including heredity and gene frequencies), and evolution at the population level, as well as the Mendelian understanding of the structure (such as chromosomes, genes, alleles) and mechanisms (such as recombination and mutation). This is the so-called new or modern synthesis of evolutionary biology.

Particle Swarm Optimization: From Jason Brownlee's *"Clever Algorithms"* - Particle Swarm Optimization belongs to the field of Swarm Intelligence and Collective Intelligence and is a sub-field of Computational Intelligence. Particle Swarm Optimization is related to other Swarm Intelligence algorithms such as Ant Colony Optimization and it is a baseline algorithm for many variations, too numerous to list. It is inspired by the social foraging behavior of some animals such as flocking behavior of birds and the schooling behavior of fish.

Firefly Algorithm: From Xin-She Yang *"Nature-Inspired Metaheuristic Algorithms"* - The flashing light of fireflies is an amazing sight in the summer sky in the tropical and temperate regions. There are about two thousand firefly species, and most fireflies produce short and rhythmic flashes. The pattern of flashes is often unique for a particular species. The flashing light is produced by a process of bioluminescence, and the true functions of suchsignaling systems are still being debated. However, two fundamental functions of such flashes are to attract mating partners (communication), and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism to remind potential predators of the bitter taste of fireflies. The firefly algorithm tries to mimic the attractiveness of Fireflies and has three basic rules:

- All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex;
- Attractiveness is proportional to the their brightness, thus for any two flashing fireflies, the less brighter one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly;
- The brightness of a firefly is affected or determined by the landscape of the objective function.

Bees Algorithm: From Jason Brownlee's *"Clever Algorithms"* - The Bees Algorithm beings to Bee Inspired Algorithms and the field of Swarm Intelligence, and more broadly the fields of Computational Intelligence and Metaheuristics. The Bees Algorithm is related to other Bee Inspired Algorithms, such as Bee Colony Optimization, and other Swarm Intelligence algorithms such as Ant Colony Optimization and Particle Swarm Optimization. It is inspired by the foraging behavior of honey bees. Honey bees collect nectar from vast areas around their hive (more than 10 kilometers). Bee Colonies have been observed to send bees to collect nectar from flower patches relative to the amount of food available at each patch. Bees communicate with each other at the hive via a waggle dance that informs other bees in the hive as to the direction, distance, and quality rating of food sources.

II. Metodology

A. Optimimzation Problem Selection

The two problems selected for this paper were *Spring Tension Design* and *Pressure Vessel Design*. Although it was simple to choose the first two problems from the computational work statements, the choice was more than justified because these aroblems are well known in the literature. Thus, the problem selection was driven by which has more that one source to compare results.

B. Programming Language

The chosen programming language for the test code was *Python Language*, version 3.10, because it is an opensource language easy to program and has a huge amount of packages regarding artificial intelligence, genetic algorithms and swarm based algorithms. From these packages it was selected *MealPy* package, version 2.5.1, because it comprises all the algorithm's classed tested in this paper.

C. Algorithm Selection

The algorithm selection was made in two steps: first, it was extracted simple version of the two classes (evolutionary and swarm based) and then it was used a simple shuffle using

D. Model Parameters

For all models and problems, were selected the following parameters:

- *Runs*: 100 runs
- *Epochs*: 100 epochs
- *Population*: 100 initial points

The initial solutions were randomly selected using the same seed for all algorithms.

E. Model Tuning

III. Results

A. Spring Tension Design

1) Evolutionary Algorithms:

2) Swarm Based Algorithms:

B. Pressure Vessel Design

1) Evolutionary Algorithms:

2) 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.