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. The algorithms were randomly chosen from a vast list of *MealPy*'s 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 standard python functions, 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/or problem solved and the Friedman's chi squared test for similarity make it noticeable because, although the values for best fits are similar, running the same algorithm with different initial conditions does not converge to similar values.**

***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. Methodology

A. Optimization 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 than one source to compare results.

1) Pressure Vessel Design: From *Solving Design of Pressure Vessel Engineering Problem Using a Fruit Fly Optimization Algorithm - XIANTING KE et al* - A pressure vessel design model involves four decision variables: x_1 is defined thickness of the pressure vessel T_s , x_2 stands for thickness of the head T_H , x_3 represents inner radius of the vessel R , and x_4 is on behalf of length of the vessel barring head L , the total variables described as (x_1, x_2, x_3, x_4). The objective function of the problem is to minimize the total cost, including the cost of material, forming, and welding. The general pressure vessel design optimization model is expressed as:

$$\begin{aligned} f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\ +3.1661x_1^2x_4 + 19.84x_1^2x_3 \end{aligned} \tag{1}$$

subject to:

$$\begin{aligned} g_1(x) &= -x_1 + 0.0193x_3 \leq 0 \\ g_2(x) &= -x_2 + 0.00954x_3 \leq 0 \\ g_3(x) &= -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\ g_4(x) &= x_4 - 240 \leq 0 \end{aligned} \tag{2}$$

The original bounding limits of the variables, extracted from computational work statements, are:

$$\begin{aligned} 0 &\leq x_1 \leq 99 \\ 0 &\leq x_2 \leq 99 \\ 10 &\leq x_3 \leq 200 \\ 10 &\leq x_4 \leq 200 \end{aligned} \tag{3}$$

For some reason these settings does not work at all with the selected optimizers, so some research in the literature *Solving Design of Pressure Vessel Engineering Problem Using a Fruit Fly Optimization Algorithm - XIANTING KE et al* suggest the following bounding limits:

$$\begin{aligned} 0.0625 &\leq x_1 \leq 99 \times 0.0625 \\ 0.0625 &\leq x_2 \leq 99 \times 0.0625 \\ 10 &\leq x_3 \leq 200 \\ 10 &\leq x_4 \leq 200 \end{aligned} \tag{4}$$

But these settings produce many random, bizarre and noisy results in all selected optimizers. Then a proud-and-lame-homemade set of variable boundings comes in handy, obtained by tweaking the original boundings:

$$\begin{aligned} 0.75 \leq x_1 \leq 0.8 \\ 0.35 \leq x_2 \leq 0.4 \\ 39.5 \leq x_3 \leq 41.0 \\ 195.0 \leq x_4 \leq 205.0 \end{aligned} \tag{5}$$

It is not intended here to point out modeling errors of any kind, nor point out package errors made by the authors or ours, but the homemade bounding limits was necessary to reach the literature results.

2) Spring Tension Design: From *Nature-Inspired Metaheuristic Algorithms - Xin-She Yang* - The design of a tension and compression spring is a well-known benchmark optimization problem. The main aim is to minimize the weight subject to constraints on deflection, stress, surge frequency and geometry. It involves three design variables: the wire diameter x_1 , coil diameter x_2 and number/length of the coil x_3. This problem can be summarized as:

$$f(x) = x_1^2 x_2 (2 + x_3) \tag{6}$$

subject to

$$\begin{aligned} g_1(x) &= \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\ g_2(x) &= \frac{4x_2^2 - x_1 x_2}{12566\left(x_1^3 x_2 - x_1^4\right)} \\ &\quad + \frac{1}{5108 x_1^2} - 1 \leq 0 \\ g_3(x) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\ g_4 &= \frac{x_1 + x_2}{1.5} - 1 \leq 0 \end{aligned} \tag{7}$$

The original bounding limits of the variables, extracted from computational work statements, are:

$$\begin{aligned} 0.05 \leq x_1 \leq 2.0 \\ 0.25 \leq x_2 \leq 1.3 \\ 2.0 \leq x_3 \leq 15.0 \end{aligned} \tag{8}$$

B. Constraint Implementation

The constraints that all problems are subjected were implemented as *penalty functions*, that is, it adds a high value when the constraint is not satisfied, zero otherwise. It is an optional requirement from computational work in question and is the recommended way to put constraints from *MealPy*'s Manual.

C. 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.

D. 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 Python's *random.shuffle* in a terminal - no script was required.

The chosen algorithms were:

- ***Evolution Based Algorithms:*** LevyEP (Evolutionary Programming), OriginalES (Evolution Strategy) and BaseGA (Genetic Algorithm)
- ***Swarm Based Algorithms:*** OriginalBeesA (Bees Algorithm), OriginalFFA (Original Firefly Algorithm) and OriginalPSO (Particle Swarm Optimization)

E. Optimizer Tuning

The hyperparameters for each optimizer were tuned using the *MealPy*'s *Tuner* utility. It is a recomended procedure to tune hyperparameters for each optimizer and problem, according to *MealPy*'s Manual. The *Tuner* utility is a very simple grid search metaheuristic search tool that test each grid configuration for a specified oprimizer runs. Although simple, it is a very expensive procedure that took 2 days to complete. It was defined 10 runs for each configuration, with the following set of hyperparameter:

- ***Evolution Based Algorithms:***
  LevyEP:
  bout_size (float): percentage of child agents implement tournament selection.
  OriginalES:
  lamda (float): Percentage of child agents evolving in the next generation.
  BaseGA:
  pc (float): cross-over probability
  pm (float): mutation probability
- ***Swarm Based Algorithms:***
  OriginalBeesA:
  selected_site_ratio (float)
  elite_site_ratio (float)
  selected_site_bee_ratio (float)
  elite_site_bee_ratio (float)
  dance_radius (float)
  dance_reduction (float)
  OriginalFFA:
  gamma (float): Light Absorption Coefficient
  beta_base (float): Attraction Coefficient Base Value
  alpha (float): Mutation Coefficient

alpha_damp (float): Mutation Coefficient Damp Rate

delta (float): Mutation Step Size

exponent (int): Exponent

OriginalPSO:

c1 (float): local coefficient

c2 (float): global coefficient

w_min (float): Weight min of bird

w_max (float): Weight max of bird

F. Optimizer Parameters

For all optimizers 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.

III. Results

A. Pressure Vessel Design (Original)

In this section are presented the results for the original variable boundings of Pressure Vessel Design, as exposed in Section II-A1.

The best results for all algorithms are presented in Table I:

Table I Best Fits for Pressure Vessel Design (Original)

Algorithm	x_1	x_2	x_3	x_4	f_x
EP	0.00000000	0.00000000	40.32041055	200.00000000	0.08150806
ES	0.00000000	0.00000000	40.32554414	200.00000000	0.08161190
GA	0.00337818	0.00106498	46.44471109	129.50295637	17.06920868
BeesA	0.07617198	0.07091908	104.75191725	44.23570263	2393.70106213
FFA	0.00000000	0.00000000	40.39658670	200.00000000	0.08306113
PSO	0.00000000	0.00000000	40.31961791	200.00000000	0.08149204

While EP, ES, FFA and PSO seems to give similar results, it is noticeable that the cost function gave some discrepant results for GA and BeesA.

Additional information is given by the Table II, where are summarized statistics from all 100 results given by different starting points (populations).

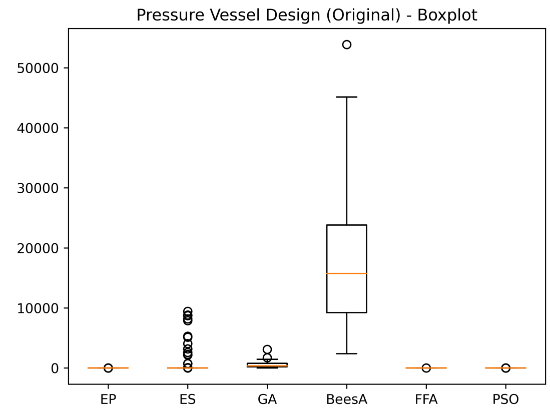
Table II Statistical Information about function values for Pressure Vessel Design (Original)

Algorithm	Min F	Mean F	Median F	Max F	StdDev F
EP	0.08150806	0.15642850	0.09093798	1.32637552	0.16807714
ES	0.08161190	718.92910990	1.00278859	9443.54269260	2060.07774212
GA	17.06920868	525.00734162	361.53879602	3112.03503082	454.52386703
BeesA	2393.70106213	17564.33024022	15751.91467117	53850.38271384	10514.60780411
FFA	0.08306113	0.13296127	0.13106271	0.20084847	0.02396876
PSO	0.08149204	0.24191732	0.08159173	10.00611523	1.00072349

From statistical viewpoint, no algorithm tested for this paper have stable solutions.

The Figure 1 gives a visual representation for Table II:

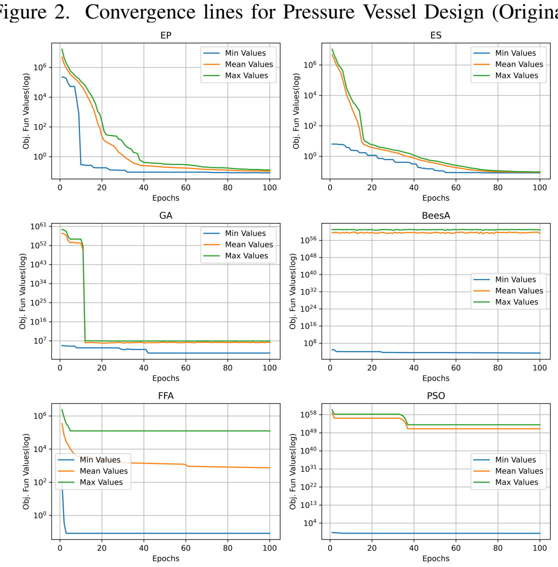
Figure 1. Boxplot for Pressure Vessel Design (Original)



Again, as it can be seen on the boxplot, there are no stability guaranteed for solution in any algorithms.

Figure 2 shows the function maximum, minimum and mean function from algorithm's evolution:

Figure 2. Convergence lines for Pressure Vessel Design (Original)



All algorithms tested have slow evolution for the original variable boundings, as it is shown in the convergence evolution.

B. Pressure Vessel Design

In this section are presented the results for a tweaked variable boundings of Pressure Vessel Design, as exposed in Section II-A1.

The best results for all algorithms are presented in Table III:

Table III Best Fits for Pressure Vessel Design

Algorithm	x_1	x_2	x_3	x_4	f_x
EP	0.75000000	0.35000000	40.68318345	195.00000000	5534.57345496
ES	0.75000000	0.35000000	40.68322151	195.00000000	5534.57917966
GA	0.75000816	0.35003661	40.68379662	195.00019653	5534.83662424
BeesA	0.75000000	0.35000000	40.68319525	195.00000000	5534.57516657
FFA	0.75000000	0.35000000	40.68318772	195.00000000	5534.57401617
PSO	0.75000000	0.35000000	40.68318348	195.00000000	5534.57345480

It is noticeable in this table that the best results and the best minimizers are quite the same, so it cannot be said that all algorithms will solve the problem from this standpoint.

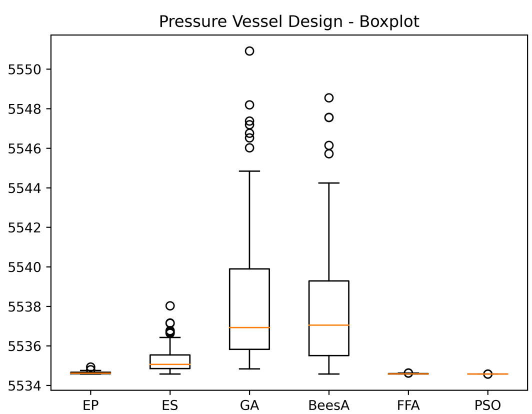
Another view of the solutions are presented in Table IV, where are summarized the statistics of all 100 possible solutions for different startpoints.

Table IV Statistical Information about function values for Pressure Vessel Design

Algorithm	Min F	Mean F	Median F	Max F	StdDev F
EP	5534.57345496	5534.63426098	5534.61416291	5534.92993607	0.05804548
ES	5534.57917966	5535.28165367	5535.06149254	5538.03003893	0.65098467
GA	5534.83662424	5538.36788889	5536.92932748	5550.91639342	3.57079951
BeesA	5534.57516657	5537.97495231	5537.04824684	5548.55587234	3.28239740
FFA	5534.57401617	5534.58849395	5534.58435493	5534.63296149	0.01341777
PSO	5534.57345480	5534.57381593	5534.57355270	5534.57625290	0.00053848

It is hard to see any statistical difference looking at any value in the table except standartd deviation. GA and BeesA seems to have the worst solutions, and EP, FFA and PSO seems to have a well defined behavior since they have the lowest standard deviations, that is, all the solutions don't spread too much from each other. A quick way to access the Table IV is shown on Figure 3.

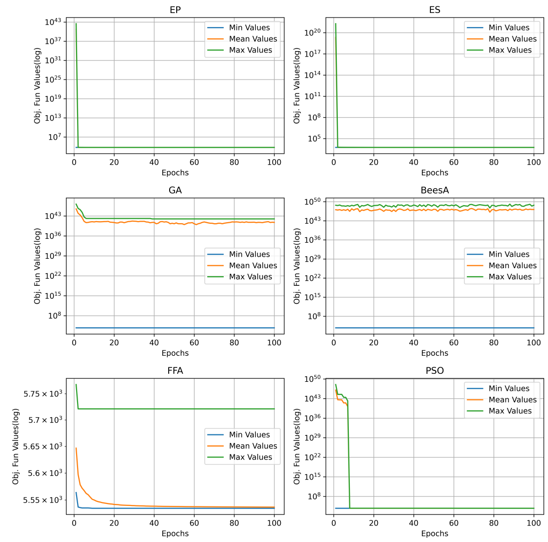
Figure 3. Boxplot for Pressure Vessel Design



It shows the same information as the Table IV and it evidences that not all algorithms have the same mean, so it is a clear evidence that not all algorithm give similar results from an arbitrary start point.

Figure 4 shows the function maximum, minimum and mean function from algorithm's evolution:

Figure 4. Convergence lines for Pressure Vessel Design



EP, ES and PSO quickly converges all the solution once it find a viable minimum, whe GA, BeesA and FFA maintain exloration and explotation for all the epochs. From Table IV , PSO has the best solution stability for our proud-lame-homemade custom bounding problem.

C. Spring Tension Design

In this section are presented the results of Spring Tension Design optimization, as exposed in Section II-A2.

The best results for all algorithms are presented in Table V:

Table V Best Fits for Spring Tension Design

Algorithm	x_1	x_2	x_3	f_x
EP	0.05078716	0.34120356	11.93931414	0.01250534
ES	0.05000000	0.32292194	13.26929635	0.01252956
GA	0.05260626	0.38419911	9.63922108	0.01252305
BeesA	0.05141008	0.35564113	11.06109433	0.01250200
FFA	0.05148403	0.35770175	10.95787946	0.01249803
PSO	0.05000000	0.32241990	13.31368880	0.01252626

Although the best fit points have noticeable differences, the cost function f_x have similar results, that is, the function f_x seems to have weak minimal points.

Another view of the solutions are presented in Table IV, where are summarized the statistics of all 100 possible solutions for different starting points (populations).

Table VI Statistical Information about function values for Spring Tension Design

Algorithm	Min F	Mean F	Median F	Max F	StdDev F
EP	0.01250534	0.01255960	0.01253156	0.01453933	0.00020190
ES	0.01252956	0.01281857	0.01274283	0.01346214	0.00024181
GA	0.01252305	0.01628575	0.01616768	0.02561808	0.00273276
BeesA	0.01250200	0.01268010	0.01261691	0.01406679	0.00021629
FFA	0.01249803	0.01254780	0.01253499	0.01268116	0.00003152
PSO	0.01252626	0.01390866	0.01318774	0.03036537	0.00342240

Figure 5. Boxplot for Spring Tension Design

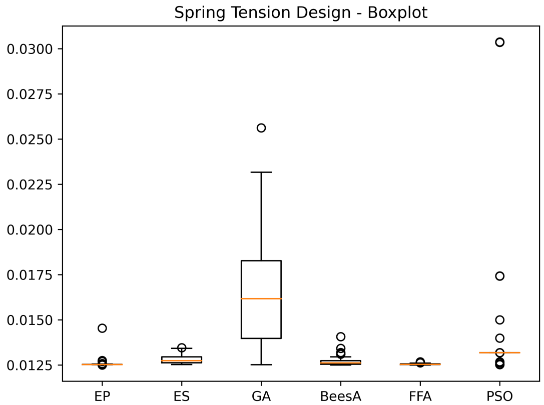
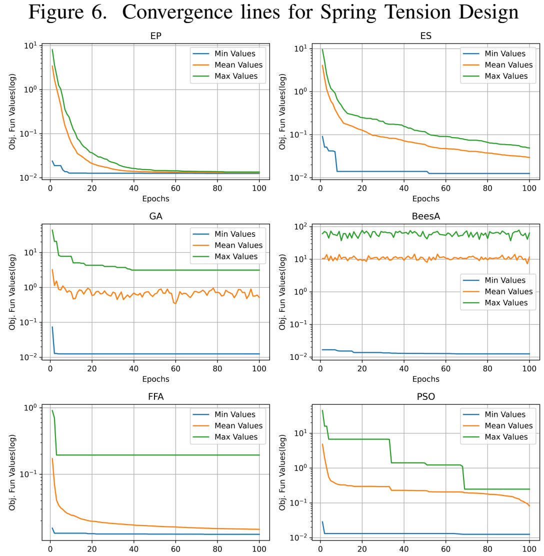


Figure 6. Convergence lines for Spring Tension Design



D. *Significance Test*

Table VII Significance Test Using Friedman Chi-Squared Test

Problem	Statistics	p-value
Spring Tension Design	339.05714286	0.00000000
Pressure Vessel Design	436.16000000	0.00000000
Pressure Vessel Design (Original)	425.07428571	0.00000000

IV. Discussion

Text goes here......