Comparison of different genetic metaheuristics with constraint techniques

1st Jose Daniel Lopez Hernandez
School of Science and Engineering
Tecnologico de Monterrey
Puebla, Mexico
josedaniel2499@gmail.com

***Abstract*—This paper will show a brief comparison of different evolutionary algorithms which aim to approximate solutions to constrained and unconstrained functions. Some of the employed meta heuristics in this paper have constraint handling techniques, with a new one being presented which worked better than expected, one of the algorithms however will be implemented without constraint handling techniques for comparison with the others for unscontrainted objective functions. furthermore, all of the algorithms are compared using the Wilcoxon rank-sum test and other more direct metrics such as the minimal value they reach.**

***Index Terms*—Metaheuristic, Wilcoxon, GA, constraints**

I. Introduction

Most problems are solvable, but even if they are, the computational expense of solving it could be astronomically large. This is why approximation methods have surfaced, whose objective is not to obtain an optimal result, but instead get as close to it as possible. Even when not having the exact result to a problem, the approximation can still normally be used for whatever you need the answer for. This approximation methods are called heuristics, however heuristics are pretty limited in their approach and they normally only use simple logic with a random approach to find the solution. This is why the next step of meta-heuristics was made. Meta-heuristics are commonly known as heuristics to choose heuristics, and this change their behaviour depending on the results the heuristics give [1]. One example of Meta-heuristics is the Genetic algorithms, Genetic algorithms create a wide array of solutions at the same time, and by analysing which solutions are better and more ”fit” to correctly approximate the solution of the problem, new solutions are created with similar features to these ”fit” solutions. This is done as many times as the user wants and usually approximates the solution better than simple heuristics [2].

II. Background

A. Genetic Algorithms

Genetic algorithms are algorithms that create individual solutions called population, or phenotype. These solutions then are evaluated as potential solutions to the problem, and depending on their correctness are assigned a fitness value. By finding the best individuals of a population, these can be reproduced, by creating new individuals that are similar to the best of the last generation, a solution can be approximated. However this could easily fall into local solutions, so by using a mutation parameter, exploration increases and global solutions are easier to find [3].

Identify applicable funding agency here. If none, delete this.

B. Constrained algorithms

Some functions include constraints in them, be it boundaries for the individual solution variables, or specific solution spaces that must be met. These constraints represent real life problems that include price, space, complexity or similar problems that can affect a solution viability. It is because of this that some minimization or maximization problems must include consideration for these, and failing to meet these constraints can be severely punishing for meta-heuristics [4].

C. Swarm algorithms

Swarm algorithms are similar to genetic algorithms in that each generation every individual has a new value. However, instead of breeding new individuals based on parents, swarm algorithm moves individuals as if they were position vectors changing with a speed vector. Every iteration of the algorithm velocity is updated to point individuals into the direction of the best found solution. This means that swarm tends to be a bit affected by local solutions if left with its original interpretation.

III. Description of the proposals

A. Genetic Algorithm

The genetic algorithm to be applied for this comparison has the following characteristics:

- **Selection**: Random Selection, individual to be bred are not chosen by any particular method, but instead they are randomly grabbed from the population (fitness is irrelevant)
- **Crossover**: Using Simulated Binary Crossover (SBX)
- **Mutation**: Using Polynomial mutation
- **Constraint handling**: Using a new method proposed in subsection III-B

B. Proposed constraint handling technique

The new constraint handling technique to be used solely in the Genetic Algorithm consist of grading the solutions. Instead of applying a direct penalty related to how many and how severe constraints are failed by, this new method acts on absolutes. Failing a constraint means that that solution will lose fitness depending on how many are failed. First a grade, G, is calculated depending on the number of constraints n, and failed constraints n_f as shown in equation 1

$$G = 1 - (n_f * \frac{1}{n}) \tag{1}$$

This grade then affects the fitness f_i of the result of the individual in question as shown in equation 2. As grade can only be decimal in value, fitness will always suffer a penalty, this also affects the result harsher if more constraints are failed.

$$f_i = f_i * G^4 \tag{2}$$

Furthermore, this proposition also takes a bit of inspiration from sudden death constraints, where failing them regardless of severity, will mean having a fitness of 0. One important point they make for failed constraints, is that they remake the specific individual which failed them, the implementation of this new method also remakes individuals if they fall below a certain grade. For the purposes of this paper the failing rate will be below .1 as the constraints in the incoming problems are very harsh.

C. Differential Evolution

The differential Evolution algorithm to be applied for this comparison will include the normal methods for it, however as it does not include a constraint handling term, the chosen method for this will be stochastic ranking [5]. The stochastic ranking will be added after the fitness evaluation and will use grading as shown in equation 1 to decide instead of the penalization factors, but results should be largely unaffected.

D. Swarm implementation

In this case, the swarm implementation will use Global best techniques, meaning that the best recorded individual will act as a global best, and each particular vector will keep track of the best result it has had. Furthermore, velocity will be updated on accounts of both of these terms, and c1 and c2 will be left as .3 and .9 respectively.

IV. Results

A. Comparison of the GA and DE algorithms

This comparison will be made on 4 of the problems showed in [4], namely G1,G4,G5 and G6. 30 runs of each algorithm will be made, with 100 population each and 20 generations. In table I the mean and standard deviation will be from the best fitness individual of each run of the algorithm. The best value refers to the best fitness individual between all 30 runs.

To further compare the algorithms a Wilcoxon rank-sum [6] comparison will be made. Table II shows statistical values of

	G1	G4	G5	G6
GA mean	-35.620	-19732.306	233.773	-7906.119
GA std	28.668	152.569	209.978	291.526
GA best val	-77.839	-20678.998	0.153	-6675.379
DE mean	-283.090	-21739.063	24.649	-7194.590
DE std	0.656	29.247	94.980	390.840
DE best val	-300.456	-22099.265	0.084	-7834.97

TABLE I
COMPARISON OF ALGORITHMS

	G1	G4	G5	G6
Function value statistic	5.41	5.41	4.67	-4.89
grades statistic	5.41	5.41	.811	0.0

TABLE II
STATISTICAL VALUE OF WILCOXON RANK-SUM

the comparison made, the first list inputted to the function was the GA output results, while the second list was the DE output results. This means that high positive values (higher than 3) means that values in the GA were higher, while high negative values (lower than -3) means that values in the DE were higher. Due to this we can interpret from the table that, while G1, G4 and G5 had worse minimization in the case of the GA they were better at respecting constraints, meaning that their values are more fit as results, even if they are higher. On the other hand G6 had more minimization by the GA while both algorithms had the same grades overall, meaning that the GA is strictly better in this case.

B. Swarm on Layeb algorithm results

As swarm was implemented without constraint handling, new algorithms were used for testing the swarm optimization method, namely the Layeb algorithms presented in [7]. The results were satisfactory, but some of the harder functions demostrated to be tough for the algorithm, namely Layeb10 and Layeb15 as shown in table III.

C. Comparison of GA, DE and Swarm Algorithms

Since Layeb algorithms do not include constraints, they will be used for the next comparisons . All of the test were made with 100 generations of 100 population. Starting with the Wilcoxon rank-sum comparison as shown in table IV. Surprisingly enough, even with the random selection the genetic algorithm got the best results, as scores in both swarm and DE tend to be higher, this is shown in all the functions except for Layeb05. Comparing the other two directly, Swarm has generally bigger values, with Layeb05 being the exception. These finding are supported by the graphs shown in figure 1 trough 4. Real encoding GA gives the best results through sheer luck, while the other two are much more closer. In this

	Layeb05	Layeb10	Layeb15	Layeb20
mean	-6.5937	57.998	4.99e+28	1.897
std	.2489	39.385	2.68e+29	2.75
best val	-6.851	.5377	.00002	.0002
target val	-6.907	0	0	0

TABLE III
SWARM RESULTS ON LAYEB ALGORITHMS

	Layeb05	Layeb10	Layeb15	Layeb20
GA vs DE	6.65	-6.65	-6.65	-6.65
GA vs Swarm	6.65	-6.65	-6.65	-6.65
Swarm vs DE	-5.97	2.85	6.65	4.96

TABLE IV
STATISTICAL VALUE OF WILCOXON RANK-SUM FOR THE THREE ALGORITHMS

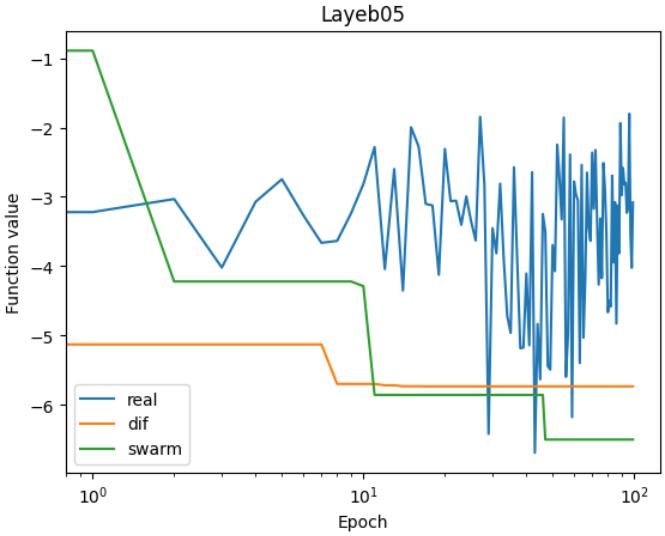

Fig. 1. Layeb05 convergence graph

specific iteration chosen, swarm beats differential evolution in two of them, while in the other it can be seen that both of them have the optimal value.

V. CONCLUSIONS

The implementation of these meta-heuristics for comparison proved that the generic genetic algorithm is the most versatile, having the best results for most of the tested functions (although this could be due to the implementation). In whichever case, the constraint handling techniques presented in this paper also showed promise, the proposed grading metric proved to be more useful that what was initially thought of it.

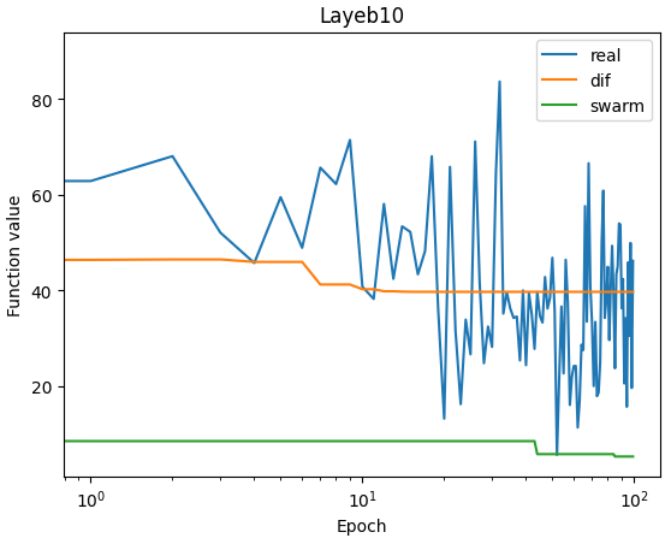

Fig. 2. Layeb10 convergence graph

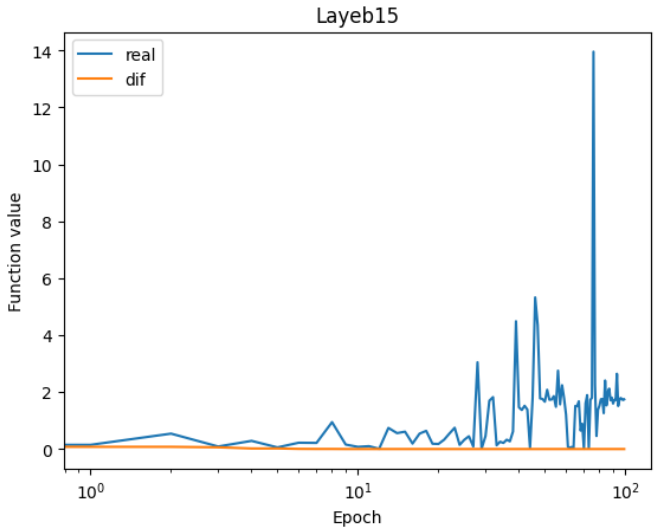

Fig. 3. Layeb15 convergence graph

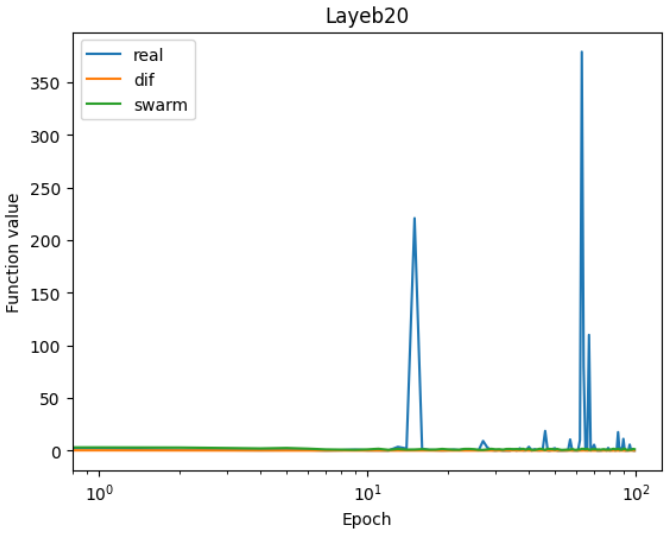

Fig. 4. Layeb20 convergence graph

REFERENCES

REFERENCES

[1] S. Desale *et al.*, "Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey," *International Journal of COmputer Engineering in Research Trends*, vol. 2, pp. 296–304, 5 2015.
[2] S. J. N. Geoffrey E Hinton *et al.*, "How learning can guide evolution," *Complex Systems*, vol. 1, pp. 495–502, 1987.
[3] K. Sastry, D. Goldberg, and G. Kendall, *Genetic Algorithms*, pp. 97–125. Boston, MA: Springer US, 2005.
[4] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, pp. 1–32, 1996.

[5] T. Runarsson and X. Yao, “Stochastic ranking for constrained evolutionary optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 284–294, 2000.
[6] C. Ford. https://library.virginia.edu/data/articles/the-wilcoxon-rank-sum-test: :text=The
[7] A. Layeb, “New hard benchmark functions for global optimization,” 2022.