Test Function Optimization

Hunter Kitts

***Abstract*—Test functions are used in multiple fields to test the robustness and performance of optimization algorithms. Because there is an increasing number of test functions, it is becoming increasingly harder to know which functions to use. The objective of this paper is to explain the testing and results of 17 different functions, including Ackley, Beale, Booth, Bukin N.6, Cross-in-tray, Easom, Eggholder, Goldstein-Price, Himmelblau's, Hölder, Lévi N.13, Matyas, McCormick, Rosenbrock, Schaffer N.2, Schaffer N.4, and Three-hump camel. The testing of these functions was achieved using an evolutionary algorithm and a partial swarm. Each test varied the parameters for the respective algorithm as to test results across a range of values. Results from this testing include an average distance to the convergence points, number of convergent tests, and a graphical representation of the testing. From the testing, it can be concluded that each test has its own strengths and weaknesses. Some tests are specifically designed to push an algorithm's optimization, whereas others optimize with little to no trouble. The testing also produced clear differences between evolutionary algorithms and particle swarms.**

I. Introduction

Determining the value of seventeen different test functions is the primary goal of the testing that was performed. However, the value of a test function can be ambiguous to determine. For instance, one user may want to test the limits of their optimization by using a function that has demonstrated to have difficulty converging. Another may want to use a test function that converges for most test cases. Because of this, each function has a different value depending on the use case. To best present this value, overarching totals for convergence have been produced. However, due to the sheer amount of data produced, only a few of the graphical representations of this testing can be included.

II. Motivation

Test functions for optimizations are becoming increasingly important in bio-inspired computing algorithms. As the complexity of these optimizations increases, so does the complexity of test functions. Currently, there are at least 50 different test functions for single-objective optimizations. This produces an environment in which researchers could have difficulty picking the correct function for their optimization. In contrast, if a chosen randomly, it could lead to a function that was "advertised as particularly challenging. [1]" As stated by Dieterich and Hartke [1] many of these functions are not made equally varying in difficulty, symmetry, and similarity to real-world applications. The intention of this project is to give empirical test values to the seventeen test functions. Specifically, for use with evolutionary algorithms and particle swarm optimizations.

III. Related Work

Most of these functions have been tested already in some capacity. In fact, they are instrumental in the testing of evolutionary algorithms. According to Bäck, two sets of benchmark functions being De Jong and Schwefel have been used to test evolutionary strategies since early inception; Schwefel's set comprises "62 functions that cover an enormous diversity of different topologies [2]." A more recent history of use can be seen from Simon 2013, with a full appendix of optimization functions, specifically with unconstrained benchmarks [3]. Again, the problem with most functions listed is their vastly different convergence rates and use cases. As described by Molga and Smutnicki Rosenbrock is a seemingly trivial valley that most optimizations can find. However, the global optimum is much more difficult to converge to [4]. The Sphere function, which has a much easier time converging in comparison, is listed directly alongside Rosenbrock in both textbooks with no indication of the strengths and weaknesses of either [2] [3].

Dieterich and Hartke tested many of these benchmark functions using global optimization [1]. The functions they tested included Ackley, Rastrigin, Schwefels, Schaffer's N.7, and Schaffer's N.6. The difference in their experiment is the different evolutionary algorithm they used. Although all the testing was done with an evolutionary approach, six different algorithms were used in their testing. Furthermore, they were able to compare the results to real world testing using the same algorithms. With this data, they found, "(a)ll benchmarks can be solved with sub-quadratic scaling, whereas in real-world applications we can get cubic scaling at best and often have to settle for much worse [1]."

In comparison, Koa and Zahara tested another seventeen different functions using a hybrid genetic algorithm and particle swarm optimization [5]. Although their testing was more directed toward the hybrid approach rather than the performance of the test functions. The hybrid between the two optimizations was concluded to be superior, as it was far more successful in the seventeen different test functions than in traditional methods. [5]." Another example of using these test functions can be found from Kennedy and Eberhart, where they only used Schaffer N.6 as a test for their particle swarm optimization [6].

Three pieces of work seem to be specifically pushing the use of these functions. The first bing Simionescu and Beale who suggested a technique for mapping these benchmark functions in a 3D space [8]. These graphs, although computationally expensive, could produce a better understanding of their impact and use cases in bio-inspired computing. The second

from Barcsák and Jármai, who recognized how evolutionary algorithms are becoming more popular and proposed a benchmark or list of test functions to be used on all optimizations. Their list comprises eight benchmark functions that have varying difficulty [9]. Finally, the last work suggested that all test functions should be used on a class of algorithms to identify their shortcomings and strengths. In this research, they identified how numerous optimizations are taking advantage of the global optimums location, bounds, and parameters [10]. To combat this, their suggestion is not only to create a benchmark but one that is novel, where the global optimum's position can be shifted and its parameters can be adjusted to create several test cases.

IV. Method

To determine the value of the seventeen different test functions, two optimization methods were used as the testing platform. The first being an evolutionary optimization and the second being a particle swarm optimization. For both testing platforms, three main libraries were used, those being LEAP, NumPy, and Matplotlib [11] [12] [13]. Each test function has a different global minimum and search domain, Table III. These values were set depending on the optimization being used. For both types of optimizations, the fitness valued would accompany a coordinate position on the test function.

As for evolutionary algorithms, the search domain was always set as the minimum and maximum values that the evolution could produce. Then the convergence point(s) were checked after each full run was completed. To create diversity in the iterations, four parameters were adjusted to test a wide range of values, Table I. Each of the parameters was tested with every combination of other parameters. This resulted in 768 tests runs for each test function. To further validate these runs, each test was run 3 times, and the values were averaged from these runs. In total, for evolutionary optimization, 2,304 tests were run for each of the 17 test functions. The maximum generation for each iteration was 30, however the local optima for each test was always reached before 30 generations. With each test, the average fitness was calculated and the best fitness value and genome were kept. Using these facts, the final point of convergence could be determined by retrieving the best values from the last generation. As stated earlier, this produced a coordinate position and a fitness value that indicated how close this value was to global optima.

TABLE I: Evolutionary Optimization

Parameters	Range
Population Size	25, 50, 75, 100
Mutation Probability	0, 0.01, 0.03, 0.05
Probability of Uniform Crossover	0, 0.1, 0.3, 0.5
Tournament Size	2, 3, 4, 5

In comparison, particle swarm optimization also varied four parameters, Table II. However, since the increments were on a much smaller level than evolutionary, the parameters have default values. So, as a range for a certain parameter was being tested, all other parameters were set to their default value. Each test ran through 1,000 generations until completion. Furthermore, the total amount of tests across the parameters was also 1,000 for each of the 17 test functions. After each run, the parameters, fitness value, and the final coordinate position. With this information, it was also determined if the optimization converged.

TABLE II: Particle Swarm Optimization

Parameters	Range	Increment	Default Value
Number of Particles	10 - 100	10	40
Inertia	0.1 - 1.0	0.1	0.5
Cognition	0.1 - 4.0	0.1	1
Social	0.1 - 4.0	0.1	1

Since both optimizations returned a fitness value and coordinate position, the next step became plotting the coordination positions along the test functions graphs. This allowed for a better visualisation of the functions convergence. To further this information, the average fitness value and convergence percentage were also returned. To determine the average distance from the convergence point or global minimum, the average fitness was subtracted.

V. Results

In Table IV, the convergence data for each function can be seen. The table includes two average columns, which detail the average distance to the convergence point(s) for each optimization. The objective for each of these averages is to be as close to 0 as possible, meaning average fitness would be that of the convergence point(s). Given this, the results show the Eggholder function had the most spread for evolutionary optimization Figure I, and the Cross-in-tray had the least amount of spread Figure II.

Fig. 1: Eggholder Evolutionary

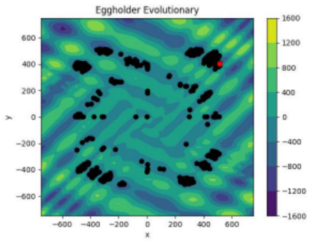
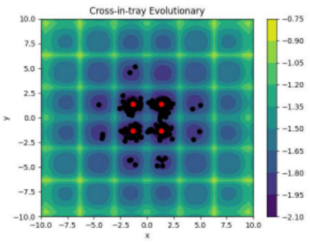


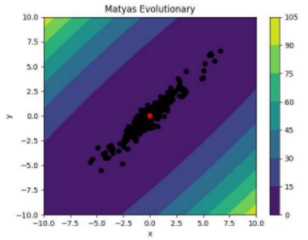
Fig. 2: Cross-in-tray Evolutionary



Convergence is slightly skewed for all evolutionary optimization functions. Unfortunately, because of local optima,

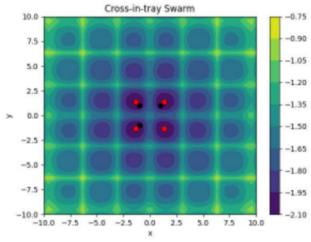
none of the test functions using evolutionary optimization reached the convergence point. Although it can be seen from Table IV, that some functions were close to the convergence point, none hit the exact value. As stated earlier, this was not because of the optimization not having enough generations. The best fitness from the optimization was usually achieved around the 16th to 19th generation. Across the board, evolutionary algorithms reach a local optima and, in most cases, could not push beyond that limit. Because of this, the percentage values have been skewed for evolutionary optimization as to better reflect the results of optimization. Thus, in terms of total convergence, Matyas had the highest percentage Figure 3, whereas several functions reached 0% convergence rate.

Fig. 3: Matyas Evolutionary



Across the included images, the black dots are the final points where the optimization landed. The number of black dots is equal to the number of total runs. The red dot(s) are the global minimum or the point of convergence. In comparison, the particle swarm optimizations had much better averages and convergence rates across the board. In terms of average fitness across all tests, Cross-in-tray performed the best. It also had the highest amount of converging tests at 99.60% Table IV. In this instance, I think it is worth comparing the two optimizations in figures. From Figure 2 and Figure 4, we can see the differences between the two optimizations.

Fig. 4: Cross-in-tray PSO



Evolutionary optimization could converge onto the global minimum, but compared to particle swarm, it is far less performant. The question then becomes whether this test is valid across either optimizations. For particle swarm optimization, is a test that almost always passes useful for benchmarking different or new methods? In some ways, yes; As long as the user understands that, it will most likely converge on all cases. In this way, if an optimization method produces lackluster results on a Cross-in-tray test, users can recognize either there is an issue with the optimization or its performance is not as well suited. If this was the only test used on a particle swarm optimization and it returned perfect results back, however, it is much harder to determine if the test resulted in any usable data. Is a test function that returns perfect data across most extremes useful by itself? Compared to evolutionary optimization, this test is average among the other functions. If an evolutionary function produced much higher results, it could be concluded that the optimization was desirable. This would be much harder to determine with a particle swarm optimization method using the Cross-in-tray function.

Fig. 5: Bukin N.6 Evolutionary

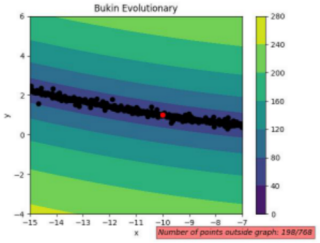
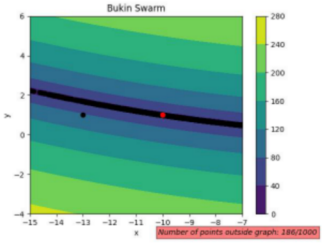


Fig. 6: Bukin N.6 PSO



An interesting parameter that has not been explored is the lack of symmetry. Beale is one example of this and the results of which can be seen in Figure 7 and Figure 8. It also has a convergence point that is not of the origin. The performance, as shown in Table IV, gives decent return for both types of optimization used as well. What gives this function more value is the lack of symmetry. An optimization can not rely on symmetry to find the point of convergence. Because of this, in both types of optimization used, the best convergence found was sometimes to the left, where the optimization would find good fitness values in the topology and had problems increasing outside of those bounds.

Most of these arguments can be made across all the test functions and there given data. However, I think given this information; it is best to just understand that some functions, although produce good results, are not necessarily as useful for benchmarking optimizations. The same can be said for difficult functions that rarely converge. These functions have a place just mainly for difficulty testing of the optimization. As seen in Figure 5 and Figure 6, Bukin is a good example of this. Both particle swarm and evolutionary optimizations find the heat on

Fig. 7: Beale Evolutionary

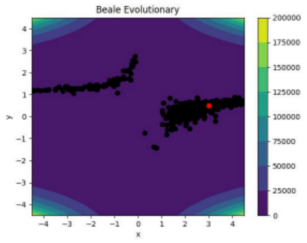
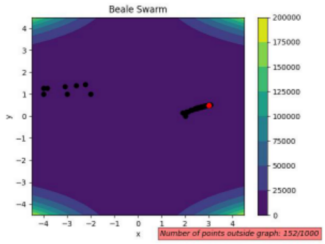


Fig. 8: Beale PSO



this function. However, it is similar to the Rosenbrock function in that the convergence point is much harder to find. The results produced on the particle swarm optimization have an average closer to the convergence point, but that is seen across all test functions.Figures 9 and 10 display the number of tests that converged for both Evolutionary and Particle Swarms, respectively. They demonstrate the strengths of particle swarms' optimizations more than anything. However, considering the limitations that evolutionary optimizations present, it is best to take these as two separate representations on the same test functions.

Fig. 9: Evolutionary Convergence

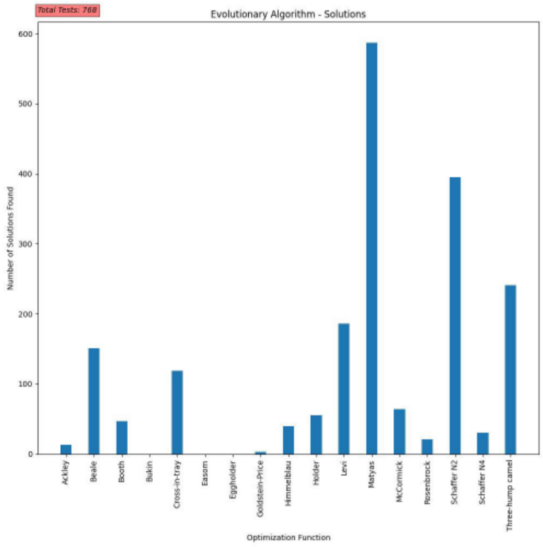
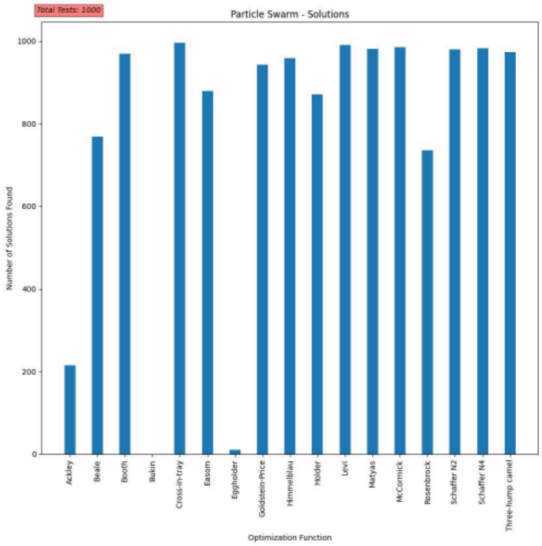


Fig. 10: Particle Swarm Convergence



VI. Conclusion

In conclusion, many of the test functions have both positive and negative attributes when used for evolutionary and particle swarm optimization methods. From the findings, I think it is safe to assume using only one test function is inadequate to benchmark and optimization method. Some of the included test functions have near perfect convergence, whereas others have difficulty ever converging. One good guideline to follow may be to use varying different levels of test functions before finalizing an optimization method. For instance, if a benchmark was needed for an evolutionary algorithm, some good tests would be Matyas, Rosenbrock, Bukin N.6, and Eggholder. These varying tests have different search domains, convergence rate, and average spread. This would allow for a spread of information to see how the optimization works in different settings. In comparison, for particle swarm optimization, the functions for Ackley, Bukin N.6, Rosenbrock, and Cross-in-tray could be used. Again, this would allow for a diverse range of testing, with some designed to converge on most of the time and some designed to most likely fail.

VII. Future Work

Various facets of future work could be done on this subject. One important value that was left out in this testing was the time to complete each run. Some formulae were complicated and during testing, it was noticed that some functions took much longer than others. Of course, the number of test functions could also be increased, possibly testing simple optimizations on an entire set of benchmark functions. However, one important study and conclusion that could be made is to categorize these functions for different optimization methods. With some categorization of methods, it would be much easier to determine if the benchmark was a good fit for the optimization. For instance, if the optimization was specifically

particle swarms, Lévi could be considered a category I, as it has a high amount of convergence and tight grouping. This would allow users to see a function and judge immediately its use cases.

REFERENCES

[1] J. Dieterich and B. Hartke, "Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization," Applied Mathematics, Vol. 3 No. 10A, 2012, pp. 1552-1564. doi: 10.4236/am.2012.330215.

[2] Bäck, T. (1996-02-15). Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. : Oxford University Press. Retrieved 11 May. 2022, from https://oxford.universitypressscholarship.com/view/10.1093/oso/9780195099713.001.0001/isbn-9780195099713.

[3] Simon, D. (2013). Evolutionary Optimization Algorithms. Germany: Wiley. https://www.google.com/books/edition/ Evolutionary_Optimization_Algorithms/gwUwIEPqk30C?hl=en&gbpv=0

[4] Molga M., Smutnicki C. (2005) Test functions for optimization needs. Available: http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf.

[5] Kao, Y.T., and Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl. Soft Comput. 8, 849–857. https://doi.org/10.1016/j.asoc.2007.07.002.

[6] Kennedy, J., & Eberhart, R.C. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942-1948 vol.4.

[7] Mishra, Sudhanshu K., Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method (August 23, 2006). Available at SSRN: https://ssrn.com/abstract=926132 or http://dx.doi.org/10.2139/ssrn.926132

[8] Simionescu, Petru & Beale, David. (2002). New Concepts in Graphic Visualization of Objective Functions. 2. 10.1115/DETC2002/DAC-34129.

[9] Barcsák, C., & Jármai, K. (2013). Benchmark for testing evolutionary algorithms. https://mae.ufl.edu/mdo/Papers/5124.pdf

[10] J. J. Liang, P. N. Suganthan and K. Deb, "Novel composition test functions for numerical global optimization," Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., 2005, pp. 68-75, doi: 10.1109/SIS.2005.1501604.

[11] Dr. Jeffrey K. Bassett, Dr. Mark Coletti, Eric Scott (2021). LEAP: Evolutionary Algorithms in Python. https://github.com/AureumChaos/LEAP

[12] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

[13] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

TABLE III: Test Functions

Functions	Global minimum	Search Domain
Ackley	$f(0,0) = 0$	$-5.12 \leq x, y \leq 5.12$
Beale	$f(3, 0.5)$ = 0	$-4.5 \leq x, y \leq 4.5$
Booth	$f(1,3) = 0$	$-10 \leq x, y \leq 10$
Bukin N.6	$f(-10,1) = 0$	-15 $\leq$ x $\leq$ 15 -3 $\leq$ y $\leq$ 3
Cross-in-tray	$f(1.34941, -1.34941) = -2.06261$ $f(1.34941, 1.34941) = -2.06261$ $f(-1.34941, 1.34941) = -2.06261$ $f(-1.34941, -1.34941) = -2.06261$	$-10 \leq x, y \leq 10$
Easom	$f(\pi, \pi) = -1$	$-100 \leq x, y \leq 100$
Eggholder	$f(512, 404.2319) = -959.6407$	$-512 \leq x, y \leq 512$
Goldstein-Price	f(0,-1) = 3	-2 $\leq x, y \leq 2$
Himmelblau	$f(3.0, 2.0) = 0.0$ $f(-2.805118, 3.131312) = 0.0$ $f(-3.779310, -3.283186) = 0.0$ $f(3.584428, -1.848126) = 0.0$	$-5 \leq x, y \leq 5$
Hölder	$f(8.05502, 9.66459) = -19.2085$ $f(-8.05502, 9.66459) = -19.2085$ $f(8.05502, -9.66459) = -19.2085$ $f(-8.05502, -9.66459) = -19.2085$	$-10 \leq x, y \leq 10$
Lévi	$f(1,1) = 0$	$-10 \leq x, y \leq 10$
Matyas	$f(0,0) = 0$	$-10 \leq x, y \leq 10$
McCormick	$f(-0.54719, -1.54719) = -1.9133$	$-1.5 \leq x \leq 4$ $-3 \leq y \leq 4$
Rosenbrock	$n = 2f(1,1) = 0$	$-\infty \leq x, y \leq \infty$
Schaffer N.2	$f(0,0) = 0$	$-100 \leq x, y \leq 100$
Schaffer N.4	$f(0, 1.25313) = 0.292579$ $f(0, -1.25313) = 0.292579$ $f(1.25313, 0) = 0.292579$ $f(-1.25313, 0) = 0.292579$	$-100 \leq x, y \leq 100$
Three-hump camel	$f(0,0) = 0$	$-5 \leq x, y \leq 5$

TABLE IV: Function Convergence

Functions	Global Minimum	EVO Average	EVO Percentage	PSO Average	PSO Percentage
Ackley	0	-2.051857006	1.69%	-0.000260813	21.50%
Beale	0	-0.977892415	19.66%	-0.019984088	76.90%
Booth	0	-3.555868967	6.12%	-0.000401676	96.90%
Bukin N.6	0	-18.40865313	0.00%	-0.047176022	0.00%
Cross-in-tray	-2.06261	-0.029366184	15.49%	2.4136E-07	99.60%
Easom	-1	-0.994431726	0.00%	-0.071779881	87.90%
Eggholder	-959.6407	-282.1126427	0.00%	-156.4235764	1.00%
Goldstein-Price	3	-35.02012694	0.39%	-0.203147192	94.20%
Himmelblau	0	-3.42820091	5.21%	-0.001290153	95.90%
Hölder	-19.2085	-3.727245664	7.16%	9.9037E+254	87.10%
Lévi	0	-0.281621247	24.22%	-0.000520951	99.00%
Matyas	0	-0.110378558	76.43%	-1.43302E-08	98.10%
McCormick	-1.9133	-0.259513875	8.33%	12.34432803	98.50%
Rosenbrock	0	-0.541654085	2.73%	-0.023368395	73.50%
Schaffer N.2	0	-0.211627551	51.43%	-0.000416341	98.00%
Schaffer N.4	0.292579	-0.075341346	3.91%	-0.000115517	98.30%
Theree-hump camel	0	-0.290908144	31.38%	-0.000899432	97.30%