Optimization Algorithm Benchmarking: A Case Study on the Bohachevsky 1 Problem (BF1)

Mustafa Sarı
Dept. of Mathematical Engineering
Yıldız Technical University
Istanbul, Turkey
mustafa.sari1@std.yildiz.edu.tr

Prof. Hale Köçken
Dept. of Mathematical Engineering
Yıldız Technical University
Istanbul, Turkey
hgonce@yildiz.edu.tr

Lect. PhD. Gökhan Göksu
Dept. of Mathematical Engineering
Yıldız Technical University
Istanbul, Turkey
gokhan.goksu@yildiz.edu.tr

***Abstract*—This report benchmarks various optimization algorithms using the Bohachevsky 1 Problem (BF1). The study evaluates the success rate, average number of iterations, and average execution time for each algorithm. Additionally, the dependency of convergence on initial conditions and the effects of changing stopping criteria and error bounds are analyzed. Results are presented in both tabular and graphical formats.**

Keywords—optimization, benchmarking, bohachevsky 1 problem, newton raphson, hestenes stiefel, polak ribiere, fletcher reeves, gradient descent, algorithm performance

I. Introduction

Optimization problems are prevalent in various fields such as engineering [1], economics, and artificial intelligence. Finding the optimal solution efficiently and accurately is crucial for many practical applications. In this report, we focus on the Bohachevsky 1 Problem (BF1), a well-known test function in optimization literature, to benchmark and evaluate the performance of different optimization algorithms [2].

The primary objective of this study is to benchmark several optimization algorithms using the Bohachevsky 1 Problem. This includes determining the number of steps required for each algorithm to find the minimum of the BF1 function, measuring and comparing the execution times of these algorithms, and investigating the dependency of algorithm convergence on initial conditions. Furthermore, the study aims to analyze the trade-offs between execution time and convergence behavior, and evaluate the impact of varying stopping criteria and absolute error bounds on the performance of the algorithms [1][2].

The algorithms tested in this study are Newton-Raphson, Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves, and Gradient Descent. For each algorithm, we provide detailed evaluations based on multiple initial values, considering their success rates, average number of iterations, and average execution times. Additionally, the implementation details of these algorithms are discussed to provide a clear understanding of their workings and performance characteristics. The results are presented in both tabular and graphical formats to facilitate comparison and understanding.

II. The Problem

The Bohachevsky 1 Problem (BF1) is defined as:

$$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \quad (1)$$

subject to $-50 \leq x_1, x_2 \leq 50$. The Bohachevsky 1 Problem is known to have multiple local minima, which makes it a challenging benchmark for optimization algorithms. The global minimizer is located at $x^* = (0,0)$ with $f(x^*) = 0$ [3].

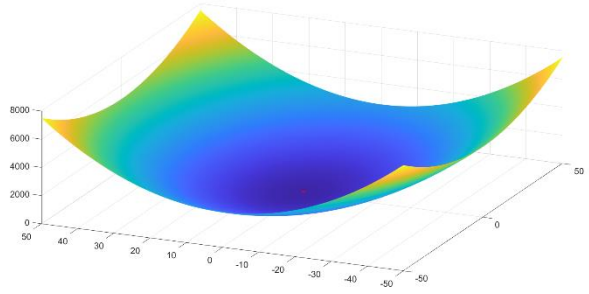

Fig. 1. Plot of the function in the domain. The red dot represents the global minima.

III. Implementation

Accurate benchmarking of the algorithms hinges on precise implementation. MATLAB was utilized for coding the algorithms and conducting the benchmarking process.

A. Setup for optimization algorithms

Initial values are selected from uniform distribution $x_{1,2} \sim \mathcal{U}_n(-50,50)$ from the problem domain interval using $rand()$ MATLAB funtion. Error bound is taken as $\epsilon = 10^{-4}$ and the stopping criterion is $\| \nabla f(xk) \| \leq \epsilon$.

B. Setup for benchmarking

In each calculation loop, three initial $x_{1,2}$ values are generated as described above, and the algorithms are executed three times per loop. To ensure precise benchmarking, each algorithm undergoes 50 loops, resulting in a total of 150 executions for different initial values. The maximum number of iterations per execution is set to 100. Upon completion of all

executions, the success rate (ratio of total number of successes to total number of executions), the average number of iterations to reach the solution and the average elapsed time are recorded. These metrics are stored to facilitate the comparison of the algorithms.

IV. Analyzing

A. Newton-Raphson Algorithm

The Newton-Raphson algorithm successfully converged in only 3.33% of the 150 executions conducted. On average, it required 99 iterations to reach the minimum, with an average execution time of 13.5941 seconds. The low success rate is primarily due to the large initial values, which often led to convergence issues or local minima. The convergence paths for different initial points are illustrated in Fig. 2.

The sensitivity to initial conditions underscores the trade-off inherent of the Newton-Raphson method. While it can converge quickly when starting near the minimum, it is less effective when the initial guess is far from the optimal point. This reliance on a good initial guess is a significant factor affecting its overall performance and success rate in our tests.

Furthermore, altering the stopping criterion and absolute error bound would impact the performance metrics. A stricter stopping criterion would likely increase the number of iterations and execution time, as the algorithm continues to refine the solution to meet more stringent conditions. Conversely, a looser stopping criterion might reduce the number of iterations and execution time, but at the cost of solution accuracy. Therefore, a balance must be maintained to optimize both computational efficiency and the accuracy of the results.

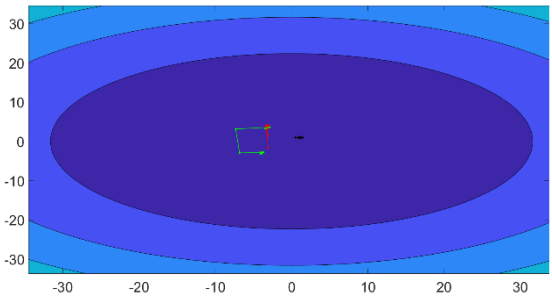

Fig. 2. Visualization of the Newton-Raphson algorithm. Each color corresponds to a different initial point, illustrating the convergence paths.

B. Hestenes-Stiefel Algorithm

The Hestenes-Stiefel algorithm demonstrated robust performance in our tests, successfully converging to the minimum in 100% of the 150 executions conducted. On average, the algorithm required only 12 iterations to reach the minimum, indicating its efficiency in navigating the optimization landscape. The average execution time was 1.0215 seconds, reflecting its computational efficiency. The convergence paths for different initial points are illustrated in Fig. 3.

The high success rate of the Hestenes-Stiefel algorithm can be attributed to its conjugate gradient method, which efficiently handles large-scale optimization problems by combining the gradient and previous search directions. This method reduces the probability of the algorithm getting stuck in local minima, thereby ensuring consistent convergence across different initial conditions.

The number of iterations and execution times are affected by the choice of stopping criterion and error bounds. Stricter criteria would likely increase both, as the algorithm would take additional steps to refine the solution. Conversely, looser criteria would result in fewer iterations and shorter execution times but might compromise the solution's accuracy.

Overall, the Hestenes-Stiefel algorithm's high success rate and efficiency make it a reliable choice for solving the Bohachevsky 1 Problem, effectively balancing convergence speed and computational cost.

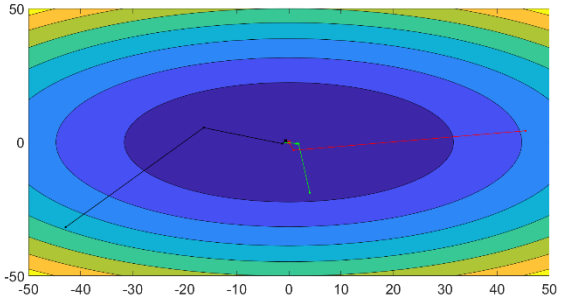

Fig. 3. Visualization of the Hestenes-Stiefel algorithm. Each color corresponds to a different initial point, illustrating the convergence paths.

C. Polak Ribiere Algorithm

The Polak-Ribiere algorithm required an average of 17 iterations to find the minimum of the Bohachevsky 1 function. The average execution time was 1.5224 seconds. Given the complexity of the problem, these numbers are reasonable and reflect the algorithm's efficiency.

The convergence of the Polak-Ribiere algorithm does not heavily depend on the initial conditions, as it successfully converged in 100% of the 150 executions conducted. This robustness is due to the algorithm's ability to adaptively adjust the search direction using information from previous iterations, which helps avoid local minima and ensures reliable convergence.

The trade-off observed is between the number of iterations and computational time. The algorithm's more complex update rules result in slightly higher execution times compared to simpler methods, but they also contribute to its robustness and consistent convergence.

Changing the stopping criterion and the absolute error bound will affect the number of iterations and execution times. Stricter criteria will increase both, as the algorithm will perform additional iterations to achieve a more precise solution. Conversely, looser criteria will decrease the number of iterations and execution time but may lead to less accurate results. Therefore, a balance must be struck between computational efficiency and the desired accuracy of the solution.

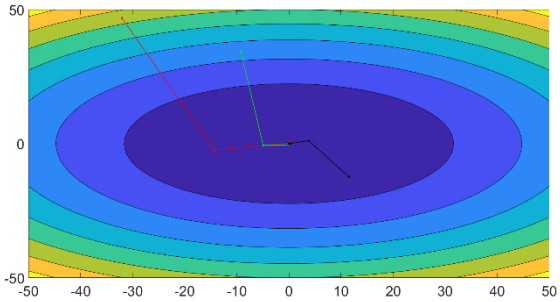

Fig. 4. Visualization of the Polak-Ribiere algorithm. Each color corresponds to a different initial point, illustrating the convergence paths.

D. *Fletcher-Reeves Algorithm*

The Fletcher-Reeves algorithm demonstrated a convergence rate of 72% across 150 executions. On average, it required 43 iterations to reach the minimum, with an average execution time of 4.1556 seconds. These metrics reflect the algorithm's balanced performance, though with a lower success rate compared to some other methods.

The Fletcher-Reeves algorithm, a variant of the conjugate gradient method, updates the search direction based on the gradient of the current and previous iterations. This method is designed to ensure that the search directions remain conjugate, promoting efficient convergence. However, its reliance on gradient information can make it sensitive to the problem's landscape and initial conditions.

The convergence rate of 72% suggests that while the Fletcher-Reeves algorithm is generally effective, it can struggle with certain initial conditions or problem characteristics, leading to more iterations and longer execution times. This is indicative of the trade-off inherent in the algorithm: its method of maintaining conjugate search directions can sometimes be less robust in navigating complex optimization landscapes.

Overall, the Fletcher-Reeves algorithm offers a moderate balance between convergence reliability and computational efficiency. Adjusting the stopping criteria and error bounds could impact its performance, potentially improving convergence rates at the cost of increased computation time.

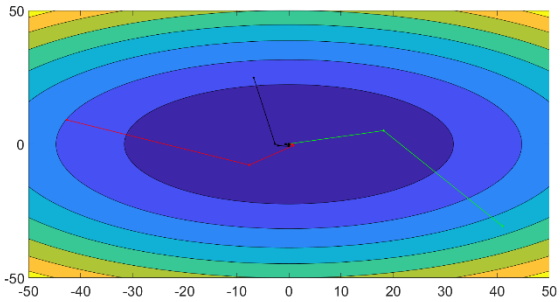

Fig. 5. Visualization of the Fletcher-Reeves algorithm. Each color corresponds to a different initial point, illustrating the convergence paths.

E. *Gradient Descent Algorithm*

The Gradient Descent algorithm was evaluated using two methods for optimizing the learning rate (α). The first method, an experimental adjustment, achieved a 100% success rate over 150 executions, averaging 57 iterations to converge with an average execution time of 5.8165 seconds. This method multiplies α by 0.9 every 5 iterations, allowing for larger initial steps and smaller, precise steps near the minimum. In contrast, the second method, backtracking line search, achieved a 96.67% success rate, averaging 31 iterations to converge with an execution time of 7.0002 seconds. This method adaptively adjusts α at each step, ensuring precise adjustments and faster convergence in terms of steps, though with additional computational overhead.

Overall, the experimental adjustment method offered quicker execution, while backtracking line search provided faster step-wise convergence but took longer due to its iterative process. The success rates and convergence iterations show both methods are effective; the experimental method is more efficient for quick results, whereas backtracking minimizes iterations. Adjusting stopping criteria and error bounds impacts performance, balancing computational efficiency and solution accuracy.

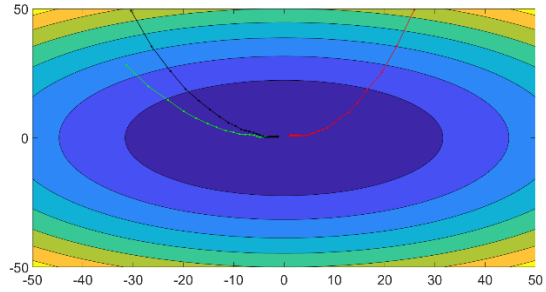

Fig. 6. Gradient Descent with experimental alpha adjustment for three initial values. Each color represents a different starting point.

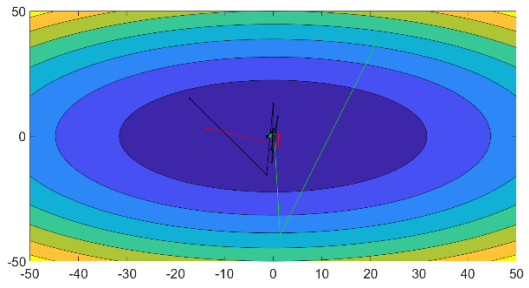

Fig. 7. Gradient Descent with backtracking line search for three initial values. Each color represents a different starting point.

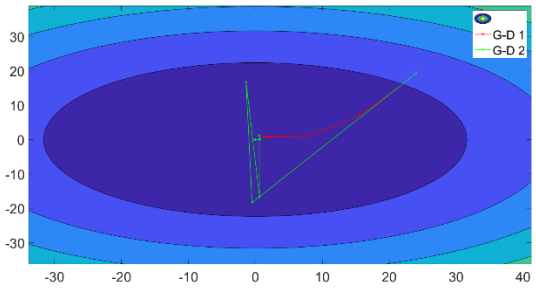

Fig. 8. Benchmarking of experimental (red) and backtracking line search (green) methods.

V. Benchmarking

To comprehensively evaluate the performance of the various optimization algorithms, a benchmark was conducted over a total of 150 exections. The results, including success rates, average number of iterations to convergence, and average execution times, are summarized in Table I and illustrated in Fig. 8.

TABLE I. Benchmarking of Optimization Algorithm

Algorithm	Success Rate (%)	Avarage Iterations	Avarage Time (s)
Newton-Raphson	3.33	99	13.5941
Hestenes-Stiefel	100	12	1.0215
Polak-Ribiere	100	17	1.5224
Fletcher-Reeves	72	43	4.1556
Gradient Descent (Experimental Adjustment)	100	57	5.8165
Gradient Descent (Backtracking Line Search)	96.67	31	7.0002

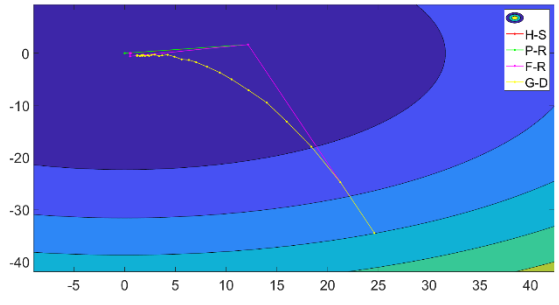

Fig. 9. Benchmarking of the optimization algorithms over 150 executions.: H-S for Hestenes-Stiefel, P-R for Polak-Ribiere, F-R for Fletcher-Reeves, G-D for Gradient Descent with experimental adjustment.

VI. Conclusion

In this study, various optimization algorithms were benchmarked using the Bohachevsky 1 Problem to evaluate their performance in terms of success rates, average number of iterations to convergence, and average execution times. The algorithms tested included Newton-Raphson, Hestenes-Stiefel, Polak-Ribiere, Fletcher-Reeves, and Gradient Descent with two different methods for learning rate adjustment [1][2].

The Newton-Raphson algorithm exhibited rapid convergence but was highly sensitive to initial conditions, resulting in a low success rate. The Hestenes-Stiefel and Polak-Ribiere algorithms demonstrated robust performance with high success rates and efficient convergence, underscoring their effectiveness in navigating complex optimization landscapes. The Fletcher-Reeves algorithm, while generally effective, showed variability in convergence depending on the initial conditions.

The Gradient Descent algorithm was evaluated using both experimental alpha adjustment and backtracking line search methods. The experimental adjustment method provided quicker overall execution, while the backtracking line search achieved faster step-wise convergence at the cost of increased computational time.

Overall, the results indicate that the choice of optimization algorithm and its specific implementation significantly impact performance. The findings highlight the importance of considering both convergence speed and computational efficiency when selecting an optimization method. Future work could explore hybrid approaches and adaptive techniques to further enhance the robustness and efficiency of optimization algorithms.

By providing a comprehensive benchmark of these algorithms, this study offers valuable insights into their practical application and sets the groundwork for further research in optimization techniques.

VII. References

[1] E. K. Chong and S. H. Żak, *An Introduction to Optimization*, vol. 75. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[2] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimisation problems," International Journal of Mathematical Modelling and Numerical Optimisation, vol. 4, no. 2, pp. 150-194, 2013.

[3] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *Journal of Global Optimization*, vol. 31, pp. 635-672, 2005.

VIII. Appendices

For complete access to the project files, including the implementation of all algorithms and additional resources, please visit the GitHub repository:

https://github.com/MustafaSari1/optimization-project