CENTRAL WASHINGTON UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCE

CS471 Project 2

Zack Lee

February 5, 2026

INTRODUCTION

This Project used ten selected standard benchmark functions of different properties. The functions were as follows: Schwefel's, 1st De Jong's, Rosenbrock, Rastrigin, Griewangk, Sine Envelope Sine Wave, Stretched V Sine Wave, Ackley's One, Ackley's Two, and Egg Holder.

To compute these functions, a matrix R^{nxm} is used. n is a constant value of 30, which is the number of experiments. m is the given dimension, which can be 10, 20, or 30, but the results in this documentation will be with the dimension of 30. To fill this matrix with values, the Mersenne Twister pseudo-random number generator is used to create the vectors (i.e. the rows in the matrix).

This project 2 builds on project 1. In this project we now use three algorithms to compute and find the best fitness and vector values. We accomplish this by using Blind Search, Local Search, and Repeated Local Search. Blind Search accomplishes its goal of finding its best vector and fitness by using the Mersenne Twister to generate pseudo-random numbers which then blind search goes through each matrix and finds the best vector and its fitness after the matrix goes through each function. Once it finds the best fitness and vector (closest to the global optima) it then stores its runtime and values in a CSV file. Next, local search works in a similar fashion by using the Mersenne twister and then based on the initial solution, will generate a new set of neighboring solutions in order to find the best fitness and vector values, if a new better solution is found it replaces the default solution, if not then the default or previous solution remains. This process is then repeated until we know that there are no better surrounding neighbors with better solutions. Repeated Local Search now implements the same process as Local Search instead now it runs the local search a repeated number

of times with only the best solutions to further optimize the solution so that the we can get closer to the global optimal

Each function will be described with its ranges, the optimal value (i.e., the most optimal fitness that we want from the function), and some of the outputs from the functions.

1 Schwefel Function

$$f_1(x) = (418.9828 \cdot n) - \sum_{i=1}^{n} -x_i \cdot sin\left(\sqrt{|x_i|}\right) \tag{1.1}$$

Schwefel function and its global optima is 0, its dimension is 30 with a range of $[-512, -512]^n$. Displayed in tables1.1 is the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 1.1: Comparative Schwefel Function Results

Blind Search

Best Fitness	Runtime (MS)
10784.0	0.0
10530.7	0.0
10728.3	0.0
9029.0	0.0
9947.3	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
8261.1	145.0
8436.6	127.7
8889.4	131.5
7982.7	164.3
7899.	130.2

2 1st De Jong's Function

$$f_2(x) = \sum_{i=i}^{n} x_i^2 \tag{2.1}$$

1st De Jong's Function and its global optima is 0: its dimension is 30 with a range of $[-100, -100]^n$. Displayed in tables2.1 is the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 2.1: Comparative 1st De Jong's Function Results

Blind Search

Best Fitness	Runtime (MS)
65543.7	0.0
66132.7	0.0
59226.5	0.0
72933.6	0.0
65839.1	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
56324.2	7.1
56353.6	17.5
49364.1	13.7
58378.9	14.0
55987.3	23.9

3 ROSENBROCK

$$f_3(x) = \sum_{i=1}^{n-1} 100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2 \tag{3.1}$$

Rosenbrock Function and its global optima is 0: its dimension is 30 with a range of $[-100, 100^n$. Displayed in tables3.1 is the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 3.1: Comparative Rosenbrock Function Result

Blind Search

Best Fitness	Runtime (MS)
30394800000.0	0.0
33920200000.0	0.0
29036700000.0	0.0
22276600000.0	0.0
27251100000.0	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
20539600000.0	23.4
23714900000.0	36.9
19271100000.0	31.1
23227100000.0	31.5
24154900000.0	34.2

4 RASTRIGIN

$$f_4(x) = 10 \cdot n \sum_{i=1}^{n} \left(x_i^2 - 10 \cdot \cos(2\pi \cdot x_i)\right) \tag{4.1}$$

Rastrigin Function and its global optima is 0: its dimension is 30 with a range of $[-30, 30]^n$. Displayed in tables4.1 is the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 4.1: Comparative Rastrigin Function Result

Blind Search

Best Fitness	Runtime (MS)
1945330.0	0.0
1757670.0	0.0
1766530.0	0.0
1622220.0	0.0
1967550.0	1.8

Repeated Local Search

Best Fitness	Runtime (MS)
1550910.0	73.4
1208000.0	73.9
1436160.0	117.3
1455570.0	78.5
1598170.0	79.3

5 GRIEWANGK

$$f_5(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{5.1}$$

The Griewangk function has a global optimum of 0. Its dimension is 30 with a range of $[-500,500]^n$. The tables 5.1 show the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 5.1: Comparative Griewangk Function Results

Blind Search

Best Fitness	Runtime (MS)
341.0	0.0
426.9	0.0
481.6	0.0
414.6	0.0
431.9	1.9

Repeated Local Search

Best Fitness	Runtime (MS)
280.2	91.1
303.0	159.4
360.9	139.1
352.1	144.8
347.9	96.7

6 Sine Envelope Sine Wave

$$f_6(x) = -\sum_{i=1}^{n-1} 0.5 + \frac{\sin\left(x_i^2 + x_{i+1}^2 - 0.5\right)^2}{\left(1 + 0.001\left(x_i^2 + x_{i+1}^2\right)\right)^2} \tag{6.1}$$

The Sine Envelope Sine Wave function has a global optimum of $-1.4915(n-1)$. Its dimension is 30 with a range of $[-30,30]^n$. The table 6.1 show the 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 6.1: Comparative Sine Envelope Sine Wave Results

Blind Search

Best Fitness	Runtime (MS)
-24.4	1.2
-25.7	0.0
-24.3	1.8
-24.1	0.0
-23.7	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
-25.6	80.8
-26.3	96.9
-26.1	128.1
-26.1	86.4
-24.9	72.5

7 Stretched V Sine Wave

$$f_7(x) = \sum_{i=1}^{n-1} \left(\sqrt[4]{x_i^2 + x_{i+1}^2} \cdot \sin\left(50 \sqrt[10]{x_i^2 + x_{i+1}^2}\right)^2 + 1 \right) \tag{7.1}$$

The Stretched V Sine Wave function has a global optimum of 0. Its dimension is 30 with a range of $[-30,30]^n$. The tables 7.1 show the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 7.1: Comparative Stretched V Sine Wave Results

Blind Search	
Best Fitness	Runtime (MS)
261.4	0.0
276.2	0.0
181.9	0.0
253.5	0.0
236.2	0.0

Repeated Local Search	
Best Fitness	Runtime (MS)
181.5	306.1
202.9	493.1
193.1	379.8
164.9	397.6
177.9	342.2

8 Ackley's One

$$f_8(x) = \sum_{i=1}^{n-1} \frac{1}{e^{0.2}} \sqrt{x_i^2 + x_{i+1}^2} + 3\left(\cos(2x_i) + \sin(2x_{i+1})\right) \tag{8.1}$$

The Ackley's One function has a global optimum of $-7.54276 - 2.91867(n-3)$. Its dimension is 30 with a range of $[-32,32]^n$. Displayed in Table 8.1 are the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 8.1: Comparative Ackley's One Results

Blind Search	
Best Fitness	Runtime (MS)
470.3	0.0
443.2	0.0
466.1	0.0
405.6	0.0
455.4	0.0

Repeated Local Search	
Best Fitness	Runtime (MS)
380.7	127.7
386.2	182.1
343.5	263.4
371.4	224.9
359.3	240.7

9 Ackley's Two

$$f_9(x) = \sum_{i=1}^{n-1} 20 + e - \frac{20}{e^{0.2\sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}}} - e^{0.5(\cos(2\pi x_i) + \cos(2\pi x_{i+1}))} \tag{9.1}$$

The Ackley's Two function has a global optimum of 0. Its dimension is 30 with a range of $[-32,32]^n$. Displayed in Table 9.1 are the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 9.1: Comparative Ackley's Two Results

Blind Search

Best Fitness	Runtime (MS)
545.6	0.0
522.3	0.0
566.9	0.0
545.3	0.0
523.5	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
470.5	406.2
494.6	367.8
490.1	380.1
474.6	351.9
476.2	407.5

10 Egg Holder

$$f_{10}(x) = \sum_{i=1}^{n-1} -x_i \cdot \sin\left(\sqrt{|x_i - x_{i+1} - 47|}\right) - (x_{i+1} + 47) \cdot \sin\left(\sqrt{\left|x_{i+1} + 47 + \frac{x_i}{2}\right|}\right) \tag{10.1}$$

The Egg Holder function does not have a closed-form global optimum. Its dimension is 30 with a range of $[-500, 500]^n$. Displayed in Table 10.1 are the first 5 best fitness findings with their runtime for each vector in milliseconds for both Blind Search and Repeated Local Search.

Table 10.1: Comparative Egg Holder Function Results

Blind Search

Best Fitness	Runtime (MS)
-2522.3	0.0
-2967.9	3.6
-3277.1	0.0
-3842.4	0.0
-3198.8	0.0

Repeated Local Search

Best Fitness	Runtime (MS)
-8292.5	199.8
-8390.4	208.8
-7018.6	253.5
-8372.5	240.1
-7192.4	192.9

11 Statistical Analysis Table With Compared Data

Table 11.1: Statistical Analysis of Blind Search Fitness Values

Function	**Average**	**Median**	**SD**	**Range**	**Total Runtime (MS)**
Schwefel	10464.7	10609.8	596.7	2108.1	12.3
1st De Jong	66648.0	67577.0	6676.6	28718.4	3.1
Rosenbrock	30705016666.7	30751500000.0	4340687192.5	17453100000.0	14.2
Rastrigin	1861848.0	1885815.0	150603.3	610870.0	19.6
Griewank	425.0	435.9	52.2	191.1	6.0
Sine Envelope Sine Wave	-23.9	-23.9	0.8	3.3	8.2
Stretched V Sine Wave	252.3	252.0	24.9	119.8	5.2
Ackley's One	458.2	455.5	30.8	146.4	12.8
Ackley's Two	544.3	545.1	12.8	45.5	17.5
Egg Holder	-3428.0	-3352.7	642.9	2503.4	19.8

[1] Surface Laptop Studio

[2] Processor 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz, 3302 Mhz, 4 Core(s), 8 Logical Processor(s)

[3] Total Ram: 2147483648 per channel 8 total

Table 11.2: Statistical Analysis of Repeated Local Search Fitness Values

Function	**Average**	**Median**	**SD**	**Range**	**Total Runtime (MS)**
Schwefel	8180.3	8251.1	384.0	1492.1	3789.9
1st De Jong	58590.1	58142.4	6262.4	27831.8	586.0
Rosenbrock	22536066666.7	23242950000.0	3763340378.1	17046200000.0	867.1
Rastrigin	1450804.0	1485460.0	165015.7	726080.0	3234.1
Griewank	327.6	328.1	36.0	145.1	3668.4
Sine Envelope Sine Wave	-25.7	-25.6	0.6	2.1	3410.2
Stretched V Sine Wave	183.3	180.3	14.7	51.6	12221.7
Ackley's One	366.4	361.9	17.0	66.1	5670.7
Ackley's Two	484.8	486.9	12.3	57.5	11510.6
Egg Holder	-7804.5	-7866.0	495.0	1741.4	6595.3

[1] Surface Laptop Studio

[2] Processor 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz, 3302 Mhz, 4 Core(s), 8 Logical Processor(s)

[3] Total Ram: 2147483648 per channel 8 total

12 Statistical Analysis With Compared Data explanation

By comparing the individual tables of each function such as Ackley's Two between the Blind Search and Repeated Local Search, we can see that the overall best fitness values are closer to the global optima compared to the Blind Search. But, we can also see that the total time to find and calculate the algorithms is larger. This amount is not by a significant amount at first glance, this is because we do the Repeated Local Search a total of 30 times, if that number increased to 1000, 10,000, or even more the time it would take to run the Repeated Local Search would just keep growing exponentially.

In my Repeated Local Search, the alpha value that I found most effective was "1.1". I tested numerous other alphas including 0.1 - 1.0 and 1.0 - 1.5, with these alphas I also tested the different amount of repetitions I wanted the Repeated Local Search to conduct and through the trial and error, I personally found that the alpha (step value) of 1.1 with a repetition of 30 provided the most optimal results without compromising the overall runtime of the function and making it overall ineffective because of the slow runtime. By increasing the alpha any further it added additional runtime because of the larger step size, and values below 1.1 were much quicker in completing the repetitions but did not provide as good of results as the alpha value that 1.1 did. So, in conclusion, the alpha value of 1.1 provided the best fitness values that attempted to reach the global optima, without compromising the overall runtime.

By comparing the statistical results of both functions, like stated before, the Blind Algorithm provides a faster runtime but a worse overall fitness, while the Repeated Local Search gives us a better fitness value and vector but a much slower execution time. So, by optimizing the Repeated Local Search and making the runtime become much faster it would be an even more effective algorithm to get the best fitness values.