CS776 Assignment1 - Black Box Hill Climber

Neal Ramaswamy

September 2025

1 Introduction

1.1 Purpose

The purpose of this assignment is to modify the Assignment0 program to find the global fitness maxima of 2 "Black Box" functions and 2 self-made functions.

2 Implementation

3 Algorithm Implementation

Initial Solution Generation The initial candidate solution is generated by randomly selecting bit values from 0, 1 with equal probability from a uniform distribution.

Neighbor Generation The neighboring solutions are generated through a bit-flip mutation where a single bit is randomly flipped (0 -¿ 1 or 1 -¿ 0).

Acceptance Criteria The algorithm adopts a greedy mindset for maximization - as soon as a better solution is found, it adopts that as the new solution and continues.

Termination Criteria The algorithm stops once it reaches the maximum number of iterations parameter.

3.1 Algorithm Psuedocode

The algorithm implemented follows a greedy-style, where it mimics a version of gradient descent where we are unaware of the true function. This is addressed through a normal distribution for generating the next gradient.

1. Generate initial binary solution by randomly setting each bit to 0 or 1 with equal probability

2. Determine the fitness of the initial solution
3. For each Iteration
   (a) Store the fitness of the current solution
   (b) Generate a neighboring solution by randomly selecting one bit and flipping it (0→1 or 1→0)
   (c) Evaluate the fitness of the neighbor solution
   (d) If the neighbor has a higher fitness, accept that as our current solution; else reject and keep the current best solution
4. Repeat this process as many times as desired

4 Function Implementation

4.1 Function Definitions

4.1.1 Blackbox Function 1

Domain: Binary vectors of length 100 (each element $\in \{0, 1\}$)
Objective: Maximize fitness (unknown internal mechanism)
Properties: External evaluation function provided as compiled object file. The algorithm considers fitness ≥ 99.9 as optimal success. The internal structure and fitness landscape are completely opaque to the optimization algorithm, representing a true black-box scenario.

4.1.2 Blackbox Function 2

Domain: Binary vectors of length 100 (each element $\in \{0, 1\}$)
Objective: Maximize fitness (unknown internal mechanism)
Properties: Second external evaluation function provided as compiled object file. Like BB1, the fitness landscape and optimal solutions are hidden from the optimizer, requiring purely exploratory search strategies.

4.1.3 Custom Easy Function

OneMax Function

$$f_{\text{easy}}(\mathbf{x}) = \sum_{i=1}^{100} x_i$$

Domain: Binary vectors of length 100 (each element $\in \{0, 1\}$)
Objective: Maximize the count of 1s in the binary string
Properties: Unimodal function with no local optima. Every bit flip from $0 \rightarrow 1$ increases fitness, making it trivial for hill climbing algorithms. Global optimum = 100 (all bits set to 1).

4.1.4 Custom Hard Function

Deceptive Trap Function

$$f_{\text{hard}}(\mathbf{x}) = \sum_{j=1}^{10} g(\mathbf{x}_{(j-1)\times 10+1:j\times 10})$$

where for each 10-bit block:

$$g(\text{block}) = \begin{cases} 10 & \text{if } \sum_{i=1}^{10} \text{block}_i = 10 \\ 9 - \sum_{i=1}^{10} \text{block}_i & \text{otherwise} \end{cases}$$

Domain: Binary vectors of length 100 (each element $\in \{0, 1\}$)
Properties: Highly deceptive multimodal function with 2^{10} local optima. Each 10-bit block creates a deceptive trap where partial solutions (1-9 ones) have lower fitness than the all-zeros state, but the all-ones state yields maximum reward. Global optimum = 100.

5 Experiment Data

Note: The red line is the average result, the green line is the highest fitness run, and the gray lines are the individual runs.

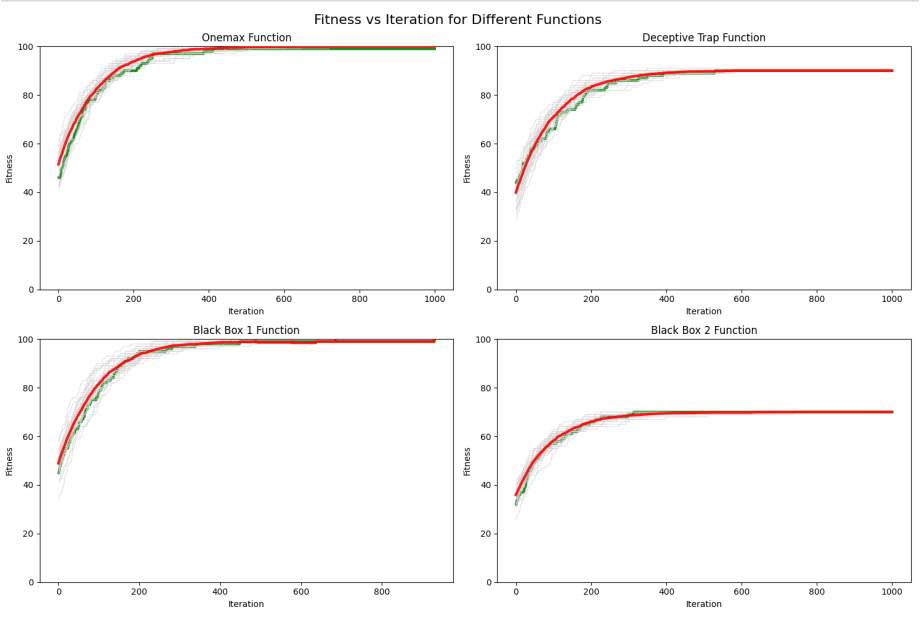

Figure 1: Comparison of different functions

6 Discussion

What is the time complexity of the hill climber? O(n)

For each iteration...

1. Vector Copying - O(n)
2. Fitness Evaluation - O(n)
3. Accepting a better solution - O(n)

Since the number of iteration is some finite constant, this results in a linear time algorithm.

What is the reliability of the hill climber? This question requires an assumption about what "reliability" means. There are three possible definitions I can think of (listing in order of likelihood):

1. How consistent is the algorithm when ran multiple times given different random starting points?
2. How generalizable is the algorithm - does it perform well on different types of optimization algorithms?
3. Does the algorithm produce identical results when given the exact same inputs and random seed?

Question 1 With a sufficient number of different starting points and assuming a relatively simple function structure, this algorithm is quite robust at finding the optimal solution. As can be seen, all 30 runs accurately found the global maximum in black box 1 and OneMax functions.

Question 2 The algorithm is not highly generalizable to complex optimization algorithms, as can be seen in the Black Box 2 and Deceptive Trap Function results; none of the 30 runs were able to find the global maximum.

Question 3 Yes, the algorithm will produce identical resulsts when given the same inputs and random seed.

How close did the hill climber get to the optimum solution? The hill climber achieved the optimal solution in OneMax and Black Box 1. For Deceptive Trap Function and Black Box 2, it was not able to find the global maximum, though through the fitness level we can see it was "closer" in Black Box 2 than deceptive trap.

What metric was used to determine how good the hill climber was? The hill climbers were assessed through "fitness." The fitness metric helps understand how close the function is to the global maximum. Assuming the maximum of these functions is known, this knowledge can be used to judge the hill climber.

References

[1] Buskulic, N., & Doerr, C. (2019). *Maximizing Drift is Not Optimal for Solving OneMax*. arXiv preprint arXiv:1904.07818.

[2] Buzdalov, M., & Doerr, C. (2021). *Optimal Static Mutation Strength Distributions for the (1+λ) Evolutionary Algorithm on OneMax*. arXiv preprint arXiv:2102.04944.

[3] Deb, K., & Goldberg, D.E. (1993). Analyzing Deception in Trap Functions. *Foundations of Genetic Algorithms 2*, 93–108. Elsevier.

[4] Deb, K., & Goldberg, D.E. (1994). Sufficient Conditions for Deceptive and Easy Binary Functions. *Annals of Mathematics and Artificial Intelligence*, 10(4), 385–408. Springer.