Assignment 3 Solution

Jacob Bodera

2024-11-10

Solution 1

Solution 2

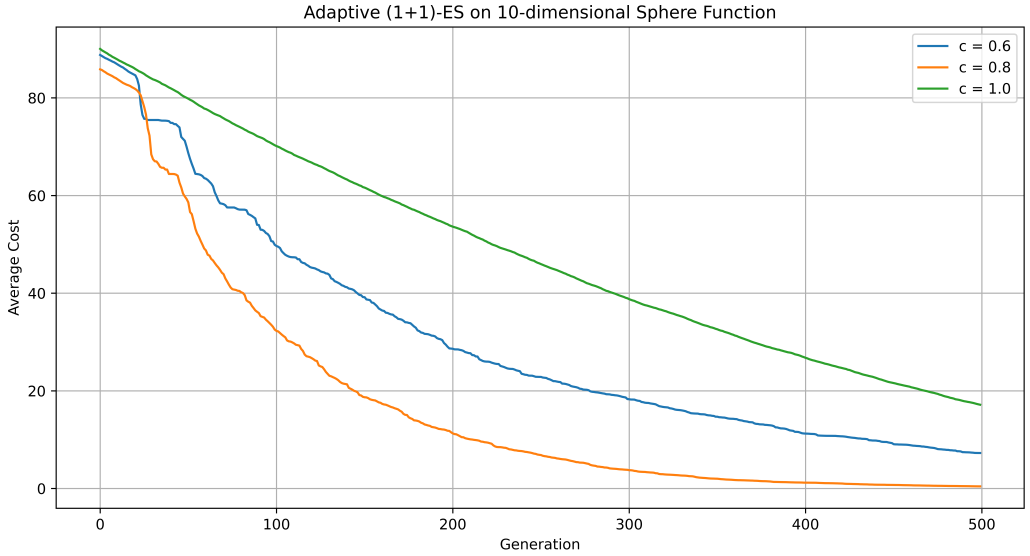
Software Implementation

```
import numpy as np
import matplotlib.pyplot as plt

def sphere(x):
    return np.sum(x ** 2)

def adaptive_es(dim, generations, c, sigma_0):
    ## Define initial x
    x = np.random.uniform(low=-5.12, high=5.12, size=dim)
    ## Set 'G' value - number of generations we look in the past to calculate success rate
    success_rate_window = 20
    ## Define some initial parameters
    sigma = sigma_0
    costs = []
    successes = []

    for generation in range(generations):
        # Apply mutation to x and bound
        x1 = x + sigma * np.random.randn(dim)
        x1 = np.clip(a=x1, a_min=-5.12, a_max=5.12)

        ## Record if mutation is successful
        if sphere(x1) < sphere(x):
            x = x1
            successes.append(1)
        else:
            successes.append(0)

        ## Update sigma based on success rate
        ## Note: if success_rate = 1/5 -- sigma = sigma
        if generation >= success_rate_window:
            success_rate = sum(successes[-success_rate_window:]) / success_rate_window
            if success_rate < 0.2:
                sigma *= c**2
            elif success_rate > 0.2:
                sigma /= c**2
        ## Append the cost of each generation
        costs.append(sphere(x))

    return costs

## Defining Testing Parameters
dim = 10
sigma_0 = 0.1/(2*np.sqrt(3))
generations = 500
num_simulations = 50
c_values = [0.6, 0.8, 1.0]

## Average costs per generation of all simulations per c value
average_costs = {c: np.zeros(generations) for c in c_values}
```

```
## For each c value, average costs over all simulations
for c in c_values:
    total_costs = []
    for _ in range(num_simulations):
        costs = adaptive_es(dim=dim, generations=generations, c=c, sigma_0=sigma_0)
        total_costs.append(costs)
    average_costs[c] = np.mean(total_costs, axis=0)

## Plot the average cost over the generation number for each c value
plt.figure(figsize=(12, 6))
for c in c_values:
    plt.plot(range(generations), average_costs[c], label=f"c = {c}")
plt.xlabel("Generation")
plt.ylabel("Average Cost")
plt.title("Adaptive (1+1)-ES on 10-dimensional Sphere Function")
plt.legend()
plt.grid()
plt.savefig("../images/problem2_a3.png", format="png", dpi=1000)
```

Result of (1 + 1)-ES Algorithm

Looking at the plot below, it appears that a value of $c = 0.8$ produced the fastest convergence of the minimization parameter $x \in \mathbb{R}^{10}$. This is consistent with the nominal value given in the problem of $c = 0.817$, however all of the cases converged fairly quickly, with latest converging model having a value of $c = 0.6$ and converging at approximately 1100 generations.



Solution 3