Implementation of Clonal selection algorithm using Python

April 8, 2025

[1]:
```python
import numpy as np

# Objective Function (Minimize)
def objective_function(x):
    return np.sum(np.square(x))  # Sphere function

# Initialize population
def initialize_population(pop_size, dim, bounds):
    return np.random.uniform(bounds[0], bounds[1], (pop_size, dim))

# Evaluate fitness
def evaluate_fitness(population):
    return np.array([objective_function(ind) for ind in population])

# Select top antibodies
def select_best(population, fitness, num_best):
    idx = np.argsort(fitness)
    return population[idx[:num_best]], fitness[idx[:num_best]]

# Clone antibodies
def clone_antibodies(selected, clone_rate):
    clones = []
    for antibody in selected:
        num_clones = int(clone_rate)
        clones.extend([antibody.copy() for _ in range(num_clones)])
    return np.array(clones)

# Hypermutation
def hypermutation(clones, fitness, beta=1.0):
    mutated_clones = []
    best_fit = np.min(fitness)
    for i, clone in enumerate(clones):
        mutation_rate = np.exp(-beta * (fitness[i] - best_fit))
        mutation = mutation_rate * np.random.randn(*clone.shape)
        mutated_clones.append(clone + mutation)
    return np.array(mutated_clones)
```

```
# Main CSA loop
def clonal_selection_algorithm(pop_size=50, dim=5, bounds=(-5, 5),␣
↪generations=10, num_best=10, clone_rate=5):
    population = initialize_population(pop_size, dim, bounds)

    for gen in range(generations):
        fitness = evaluate_fitness(population)
        best, best_fitness = select_best(population, fitness, num_best)

        clones = clone_antibodies(best, clone_rate)
        clone_fitness = np.repeat(best_fitness, clone_rate)
        mutated_clones = hypermutation(clones, clone_fitness)

        mutated_fitness = evaluate_fitness(mutated_clones)
        combined = np.vstack((population, mutated_clones))
        combined_fitness = np.hstack((fitness, mutated_fitness))

        idx = np.argsort(combined_fitness)
        population = combined[idx[:pop_size]]

        best_solution = population[0]
        print(f"Generation {gen+1}: Best Fitness = {combined_fitness[idx[0]]:.
↪6f}")

    return best_solution

# Run the CSA for 10 generations
best = clonal_selection_algorithm(generations=10)
print("\nBest solution found:", best)
```

```
Generation 1: Best Fitness = 2.578374
Generation 2: Best Fitness = 2.578374
Generation 3: Best Fitness = 2.001040
Generation 4: Best Fitness = 0.544500
Generation 5: Best Fitness = 0.544500
Generation 6: Best Fitness = 0.544500
Generation 7: Best Fitness = 0.544500
Generation 8: Best Fitness = 0.374092
Generation 9: Best Fitness = 0.250015
Generation 10: Best Fitness = 0.212501

Best solution found: [ 0.24100444  0.24463507 -0.2410191  -0.15222192
-0.11536762]
```

```
[ ]:
```