2025-02-13 Evolution Strategy with Integer Mutations on a Rotated Ellipsoid

In this study, we explore an **Evolution Strategy** that applies integer mutations to a *rotated ellipsoid* function. Our approach uses a **double geometric** (discrete Laplace) mutation operator to sample integer steps around a parent point. The ellipsoid function is evaluated on the discrete grid $\mathbb{Z}^2$, making it well-suited for problems where the variables are naturally integer-valued.

Rotated Ellipsoid Function

The rotated ellipsoid function is defined as

$$f(x) = \lambda_1 \left(x_{\text{rot},1}\right)^2 + \lambda_2 \left(x_{\text{rot},2}\right)^2,$$

where the rotated coordinates are obtained via

$$x_{\text{rot}} = R(\theta)\, x,$$

and the rotation matrix is given by

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

In our implementation, the inputs $x = (x_1, x_2)$ are truncated to integers, ensuring that the function is evaluated on a discrete grid.

Double Geometric Mutation Operator

The double geometric mutation operator samples integer steps from a two-sided geometric (discrete Laplace) distribution. The probability mass function (PMF) for a given step k is

$$P(k) = \begin{cases} \dfrac{1-p}{1+p} & \text{if } k = 0, \\[2ex] \dfrac{1-p}{1+p}\, p^{|k|} & \text{if } k \neq 0, \end{cases}$$

where $p = \exp(-1/\sigma)$ and σ is the mutation scale parameter. The mutation is applied independently to each coordinate.

Mutation and Domain Truncation

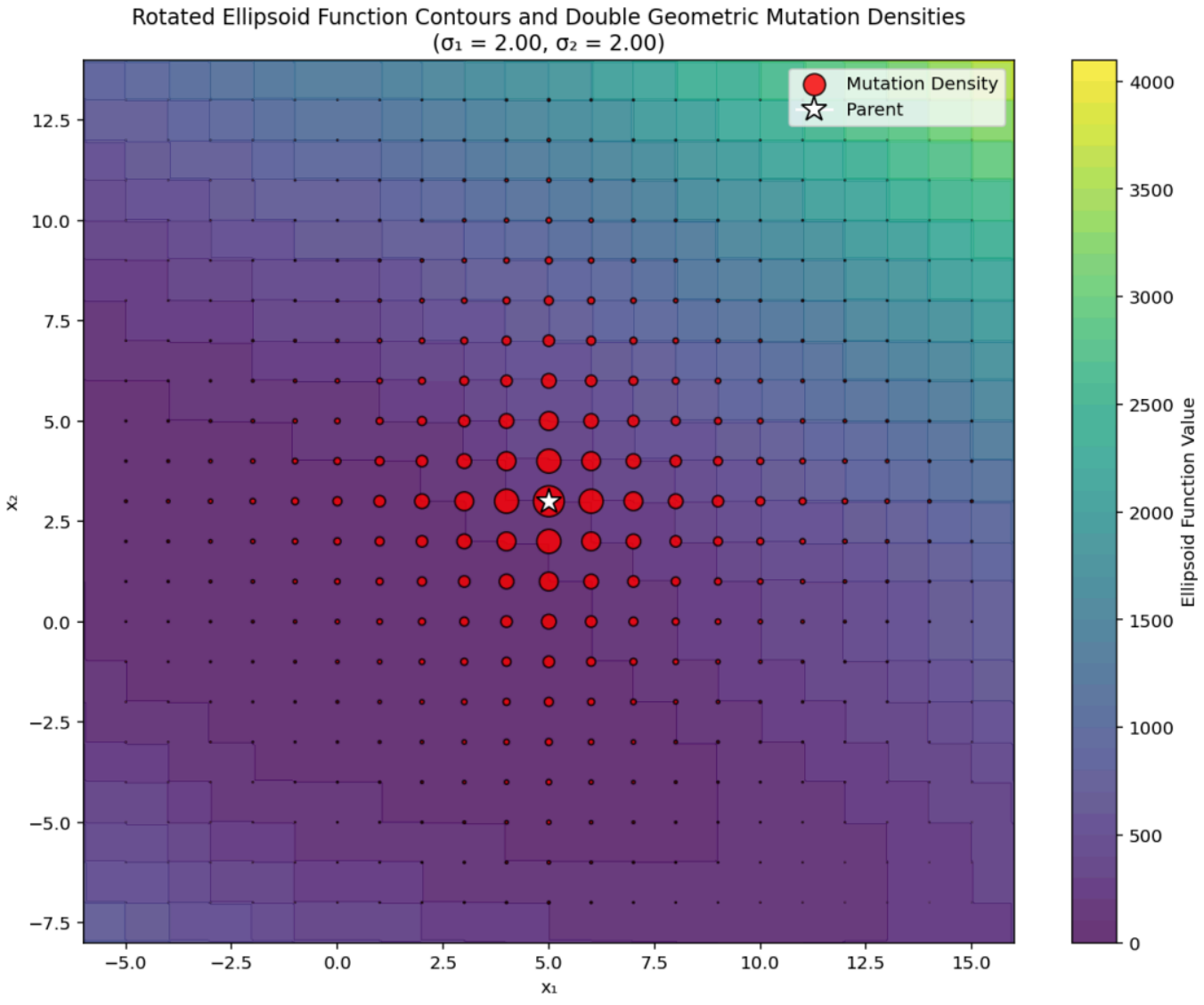
Given a parent point, for example, $x_{\text{parent}} = (7, 6)$, mutations are generated as

$$x_{\text{offspring}} = x_{\text{parent}} + (\delta_1, \delta_2),$$

with δ_1 and δ_2 drawn from the double geometric distribution. To ensure that the offspring remain within the desired search space, each coordinate is truncated to lie within the interval $[-12, 12]$.

Visualizing the Landscape

We visualize the fitness landscape by plotting contours of the rotated ellipsoid function along with the mutation density. This visualization helps in understanding the interplay between the mutation operator and the discrete nature of the objective function, and it aids in tuning the parameters for optimal search performance.




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

```python
# ------------------------------
# Rotated Ellipsoid Function (vectorized for grid evaluation)
# ------------------------------
def rotated_ellipsoid_grid(X, Y, theta=np.pi/6, lambdas=(1.0, 10.0)):
    """
    Evaluates the rotated ellipsoid function on a grid defined by X and Y.
    The inputs X and Y are truncated to integers before evaluation.
    """
    # Truncate the inputs to integers
    X_int = X.astype(int)
    Y_int = Y.astype(int)

    cosT = np.cos(theta)
    sinT = np.sin(theta)
    # Rotate the coordinates using the integer values
    X_rot = cosT * X_int - sinT * Y_int
    Y_rot = sinT * X_int + cosT * Y_int
    return lambdas[0]*X_rot**2 + lambdas[1]*Y_rot**2

# ------------------------------
# Double Geometric (Discrete Laplace) PMF
# ------------------------------
def double_geometric_pmf(k, sigma):
    """
    Returns the probability mass for a given integer step k.
    The pmf is:
      P(k) = (1-p)/(1+p)            if k == 0,
      P(k) = (1-p)/(1+p) * p^{|k|}    if k != 0,
    with p = exp(-1/sigma).
    """
    p = np.exp(-1.0/sigma)
    if k == 0:
        return (1-p)/(1+p)
    else:
        return (1-p)/(1+p) * (p ** abs(k))

# ------------------------------
# Settings and Optimal Parameters
# ------------------------------
# Parent point (can be thought of as the current solution)
x_parent = np.array([5, 3])

# Use optimal sigma parameters from your tuning procedure.
# (If not available, you can set them manually, e.g., sigma1 = 2.0, sigma2 =
2.0)
sigma1 = 2.0   # Replace with opt_sigma1 if available
```

```
sigma2 = 2.0   # Replace with opt_sigma2 if available

# --------------------------------
# Compute the Joint Mutation Distribution over Integer Offsets
# --------------------------------
max_offset = 10  # range of integer offsets in each direction for
visualization
offsets = np.arange(-max_offset, max_offset+1)

# We will compute the joint probability mass for each possible (delta1,
delta2)
mutation_points = []  # positions = parent + (delta1, delta2)
joint_probs = []      # joint probability

for d1 in offsets:
    for d2 in offsets:
        prob1 = double_geometric_pmf(d1, sigma1)
        prob2 = double_geometric_pmf(d2, sigma2)
        joint_prob = prob1 * prob2
        mutation_points.append(x_parent + np.array([d1, d2]))
        joint_probs.append(joint_prob)

mutation_points = np.array(mutation_points)
joint_probs = np.array(joint_probs)

# --------------------------------
# Prepare the Contour Plot for the Rotated Ellipsoid Function
# --------------------------------
# Define a grid covering the parent's neighborhood (to include all mutation
offsets)
x_min = x_parent[0] - max_offset - 1
x_max = x_parent[0] + max_offset + 1
y_min = x_parent[1] - max_offset - 1
y_max = x_parent[1] + max_offset + 1

xx, yy = np.meshgrid(np.linspace(x_min, x_max, 300),
                     np.linspace(y_min, y_max, 300))
zz = rotated_ellipsoid_grid(xx, yy, theta=np.pi/6, lambdas=(1.0, 10.0))

# --------------------------------
# Create the Plot
# --------------------------------
plt.figure(figsize=(10, 8))

# Plot the rotated ellipsoid function as filled contours
contour = plt.contourf(xx, yy, zz, levels=50, cmap='viridis', alpha=0.8)
```

```
plt.colorbar(contour, label='Ellipsoid Function Value')

# Overlay the mutation probability masses.
# We scale the marker size (s) by the joint probability.
# Adjust the scaling factor so the circles are visible.
scale_factor = 5000
plt.scatter(mutation_points[:, 0], mutation_points[:, 1],
            s=joint_probs * scale_factor,
            color='red', edgecolor='k', alpha=0.8,
            label='Mutation Density')

# Mark the parent point
plt.plot(x_parent[0], x_parent[1], marker='*', markersize=15,
         color='white', markeredgecolor='black', label='Parent')

plt.xlabel('x₁')
plt.ylabel('x₂')
plt.title('Rotated Ellipsoid Function Contours and Double Geometric Mutation
Densities\n'
          f'(σ₁ = {sigma1:.2f}, σ₂ = {sigma2:.2f})')
plt.legend()
plt.tight_layout()
plt.show()
```