Restarted .venv (Python 3.10.7)

```python
# Import necessary libraries
import numpy as np
from reservoirpy import nodes, datasets, observables
import hierarchical_genomes as hg
import networkx as nx
import matplotlib.pyplot as plt
import copy

# Import functions from helper file
```

```python
from expt_helper_functions import create_initial_genome,
select_best_genomes, reproduce, log_generation_results,
analyze_results

# Set up the main parameters
population_size = 100 #50
n_generations = 300  #100
mutation_probability = 0.1
insertion_probability = 0.1

# Define the number of input and output nodes for your
neural network
num_input_nodes = 10
num_output_nodes = 10

# Load the Mackey-Glass dataset with specified timesteps
n_timesteps = 1000
X = datasets.mackey_glass(n_timesteps=n_timesteps,
sample_len=1000)

elitism_factor = 0.1  # Assuming you want to carry over
10% of the population
num_elites = int(elitism_factor * population_size)

# Initialize your population
genome_population = [create_initial_genome() for _ in
range(population_size)]

# Define a fitness evaluation function
def evaluate_fitness(genome):
    # Convert genome to neural network (function from
your thesis code)
    weight_matrix =
hg.transcribe_hierarchical_genome_to_weight_matrix(genome
)

    # Setup the Echo State Network
    esn = nodes.Reservoir(W=weight_matrix) >>
nodes.Ridge(ridge=1e-6)
```

```

    # Train and forecast using the ESN
    forecast = esn.fit(X[:500], X[1:501]).run(X[502:])

    # Calculate fitness (e.g., using RMSE)
    fitness_score = observables.rmse(forecast, X[502:])
    return fitness_score

# Evolutionary loop
for generation in range(n_generations):
    fitness_scores = [evaluate_fitness(genome) for genome
in genome_population]
    selected_genomes =
select_best_genomes(genome_population, fitness_scores)

    # Update the call to 'reproduce' to include node
counts
    new_population = reproduce(selected_genomes,
population_size, mutation_probability, num_input_nodes,
num_output_nodes, num_elites)
    genome_population = new_population

    # Correct file path for logging
    log_path = "/Users/chaitravshetty/Downloads/Advanced-
Genomes-for-Evolutionary-Computing-main 3/
hierarchical_genomes/evolution_log.txt"
    log_generation_results(generation, selected_genomes,
fitness_scores, log_file=log_path)

# Perform post-experiment analysis with the correct file
path
analyze_log_path = "/Users/chaitravshetty/Downloads/
Advanced-Genomes-for-Evolutionary-Computing-main 3/
hierarchical_genomes/evolution_log.txt"
analyze_results(log_file=analyze_log_path)
```

```
Running Model-0: 500it [00:00, 11433.98it/s]
Running Model-0: 100%|██████████| 1/1 [00:00<00:00,
21.25it/s]
Fitting node Ridge-0...
```

```
Running Model-0: 498it [00:00, 13232.50it/s]
Running Model-1: 500it [00:00, 12002.45it/s]
Running Model-1: 100%|██████████| 1/1 [00:00<00:00,
21.97it/s]
```

```
Fitting node Ridge-1...
```

```
Running Model-1: 498it [00:00, 13128.70it/s]
Running Model-2: 500it [00:00, 11234.48it/s]
Running Model-2: 100%|██████████| 1/1 [00:00<00:00,
21.49it/s]
```

```
Fitting node Ridge-2...
```

```
Running Model-2: 498it [00:00, 12772.19it/s]
Running Model-3: 500it [00:00, 12176.74it/s]
Running Model-3: 100%|██████████| 1/1 [00:00<00:00,
22.79it/s]
```

```
Fitting node Ridge-3...
```

```
Running Model-3: 498it [00:00, 12176.04it/s]
Running Model-4: 500it [00:00, 12619.99it/s]
Running Model-4: 100%|██████████| 1/1 [00:00<00:00,
23.70it/s]
```

```
Fitting node Ridge-4...
```

```
Running Model-4: 498it [00:00, 11938.38it/s]
Running Model-5: 500it [00:00, 14186.43it/s]
Running Model-5: 100%|██████████| 1/1 [00:00<00:00,
26.65it/s]
```

```
Fitting node Ridge-5...
```

```
Running Model-5: 498it [00:00, 11828.72it/s]
Running Model-6: 500it [00:00, 15613.34it/s]
Running Model-6: 100%|██████████| 1/1 [00:00<00:00,
29.07it/s]
```

```
Fitting node Ridge-6...
```

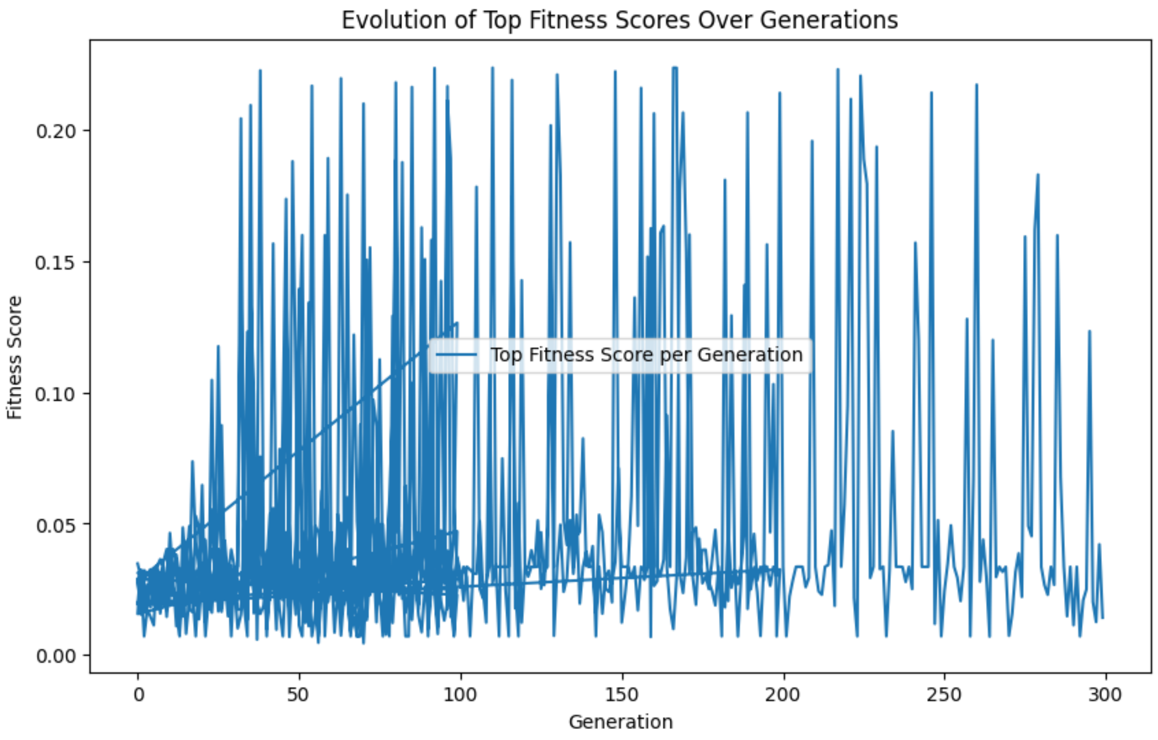
```
Running Model-6: 498it [00:00, 12469.34it/s]
Running Model-7: 500it [00:00, 15333.20it/s]
Running Model-7: 100%|██████████| 1/1 [00:00<00:00,
28.67it/s]
```

```
Fitting node Ridge-7...
```

```
Running Model-7: 498it [00:00, 12638.57it/s]
Running Model-8: 500it [00:00, 13779.01it/s]
Running Model-8: 100%|██████████| 1/1 [00:00<00:00,
25.88it/s]
```

```
Fitting node Ridge-8...
```

```
Running Model-8: 498it [00:00, 13186.47it/s]
Running Model-9: 500it [00:00, 13988.38it/s]
Running Model-9: 100%|██████████| 1/1 [00:00<00:00,
25.88it/s]
Fitting node Ridge-9...
Running Model-9: 498it [00:00, 12486.85it/s]
Running Model-10: 500it [00:00, 14976.34it/s]
Running Model-10: 100%|██████████| 1/1 [00:00<00:00,
27.47it/s]
Fitting node Ridge-10...
Running Model-10: 498it [00:00, 13341.53it/s]
Running Model-11: 500it [00:00, 15392.17it/s]
Running Model-11: 100%|██████████| 1/1 [00:00<00:00,
28.81it/s]
Fitting node Ridge-11...
Running Model-11: 498it [00:00, 12941.13it/s]
Running Model-12: 500it [00:00, 14516.68it/s]
Running Model-12: 100%|██████████| 1/1 [00:00<00:00,
26.60it/s]
Fitting node Ridge-29998...
Running Model-29998: 498it [00:00, 13057.46it/s]
Running Model-29999: 500it [00:00, 11562.84it/s]
Running Model-29999: 100%|██████████| 1/1 [00:00<00:00,
21.90it/s]
Fitting node Ridge-29999...
Running Model-29999: 498it [00:00, 13404.80it/s]
Average top fitness score across generations:
0.04364594257984812
```


Evolution of Top Fitness Scores Over Generations
Top Fitness Score per Generation
Fitness Score
Generation
0.00
0.05
0.10
0.15
0.20
0
50
100
150
200
250
300