מעבדה לבינה מלאכותית ש"י בושינסקי – כלי ויזואליזציה למנוע הגנטי

הדוגמאות הבאות בפייתון ומתבססות על הספריות הבאות

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import pandas as pd
from pandas.plotting import parallel_coordinates
from scipy.cluster.hierarchy import dendrogram, linkage
```

1. דוגמא להצגת פונקצית הפיטנס:

המטרה: מציאת מינימום לפרבולה (פ' בשני משתנים X וY)

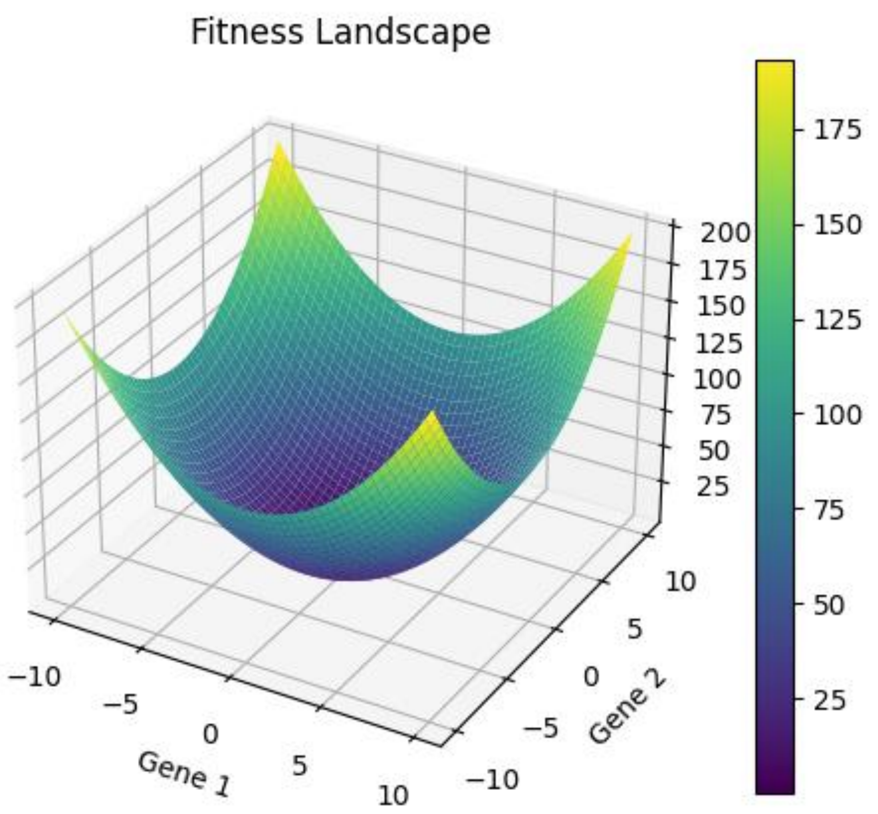
כרומוזום מורכב משני גנים

```
# Define the fitness function
def fitness_function(x, y):
    return x**2 + y**2
```

כל אלל בין 10- ל 10:

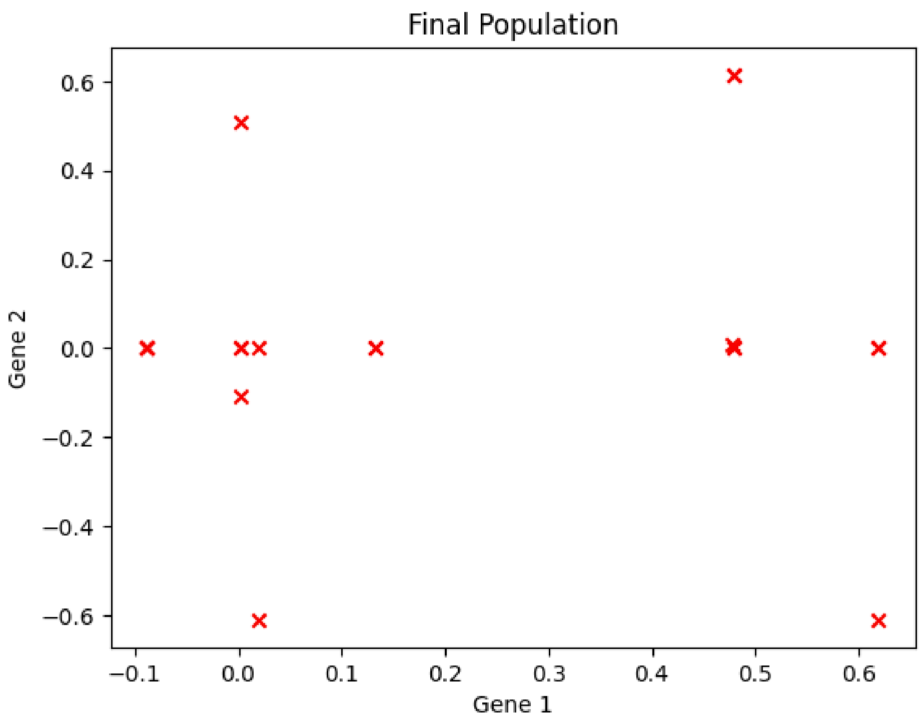
```
# Generate grid data for visualization
x = np.linspace(-10, 10, 400)
y = np.linspace(-10, 10, 400)
x, y = np.meshgrid(x, y)
z = fitness_function(x, y)

# Plot the fitness landscape
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surface = ax.plot_surface(x, y, z, cmap='viridis')
```

```
fig.colorbar(surface)
ax.set_title('Fitness Landscape')
ax.set_xlabel('Gene 1')
ax.set_ylabel('Gene 2')
ax.set_zlabel('Fitness')
plt.show()
```



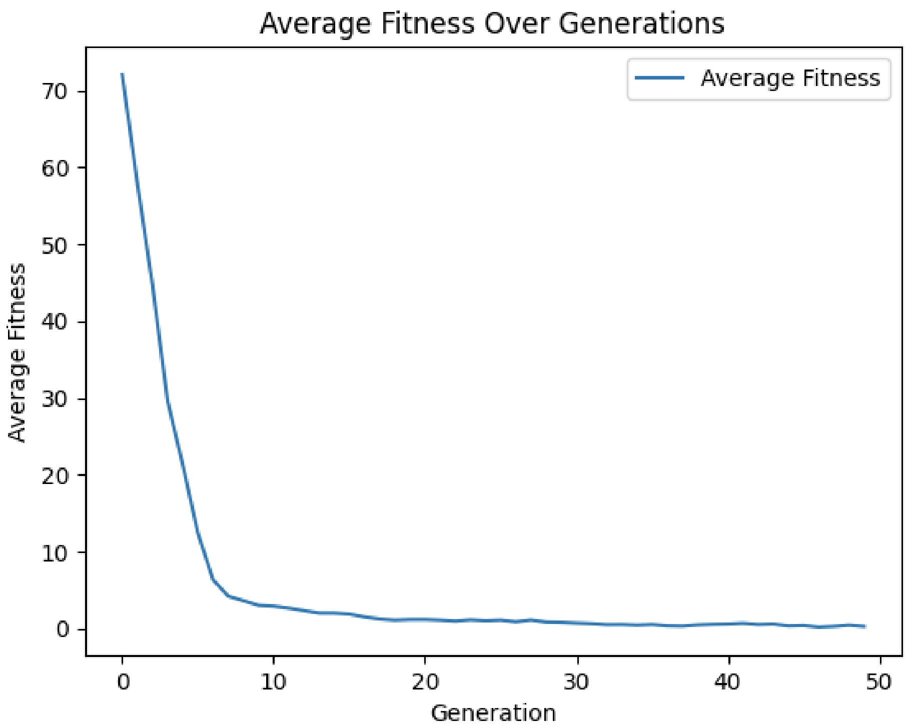
2. דיאגרמת פיזור ערכי האללים בדור האחרון

```
# Plot the final population
plt.scatter(population[:, 0], population[:, 1], c='red', marker='x')
plt.title('Final Population')
plt.xlabel('Gene 1')
plt.ylabel('Gene 2')
plt.show()
```

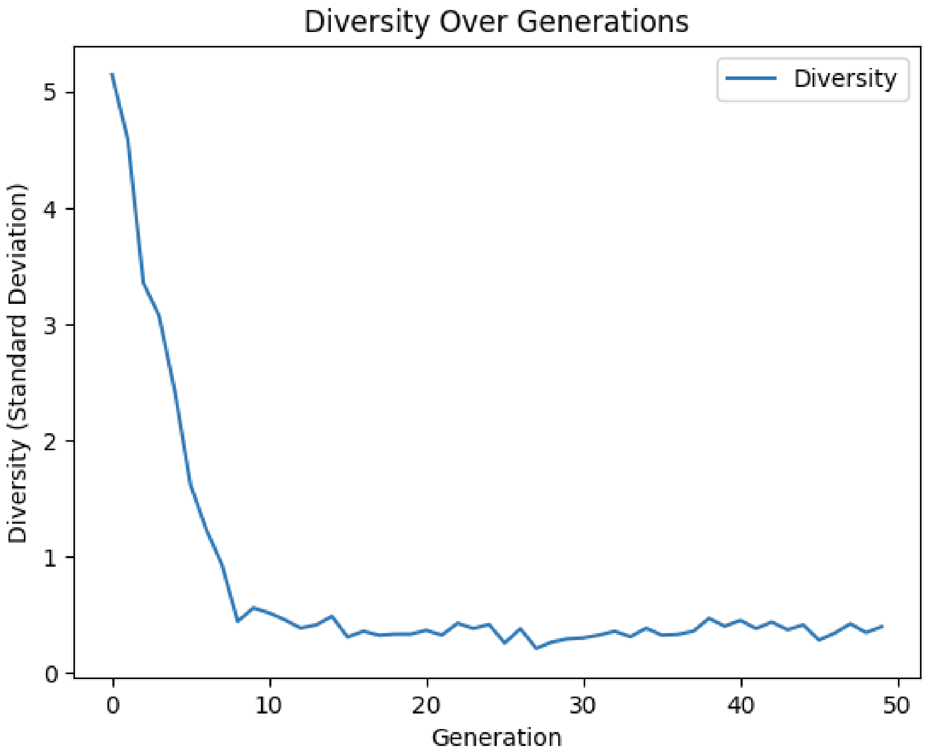


```
# 1. Average Fitness Over Generations
plt.figure()
plt.plot(range(num_generations), avg_fitness_over_time, label='Average Fitness')
plt.title('Average Fitness Over Generations')
plt.xlabel('Generation')
plt.ylabel('Average Fitness')
plt.legend()
plt.show()

# 2. Diversity Over Generations
plt.figure()
plt.plot(range(num_generations), diversity_over_time, label='Diversity')
plt.title('Diversity Over Generations')
plt.xlabel('Generation')
plt.ylabel('Diversity (Standard Deviation)')
plt.legend()
plt.show()
```

3. גרף ממוצע הפיטנס לאורך הדורות



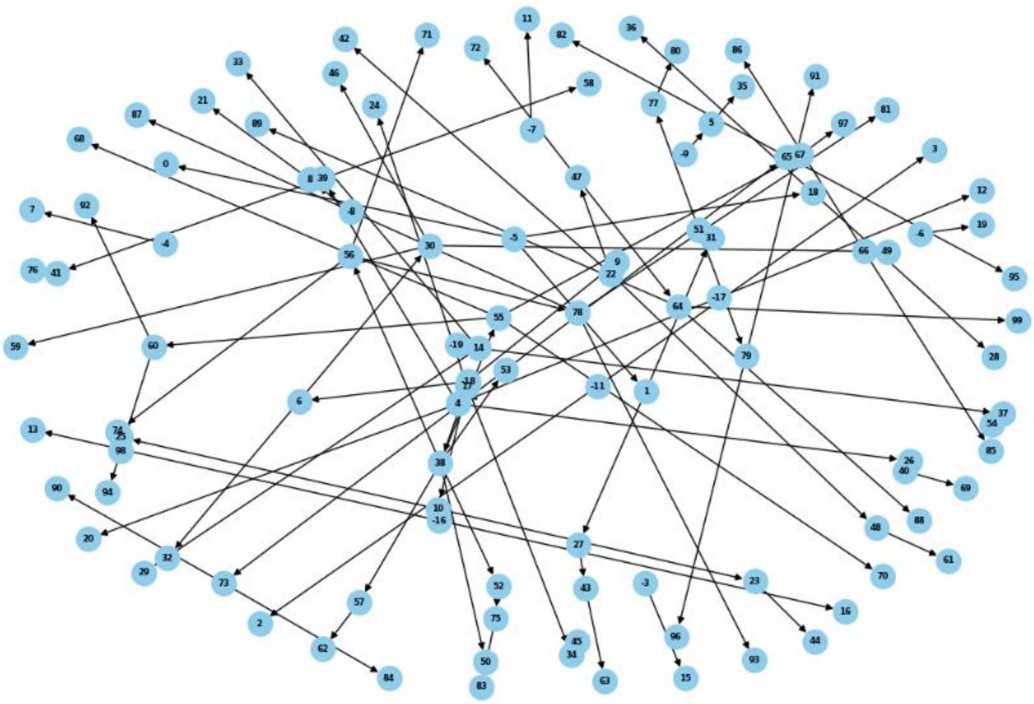
4. גרף הגוון הגנטי לאורך הדורות



5. מעקב גנאלוגי אחרי חמשת הדורות האחרונים

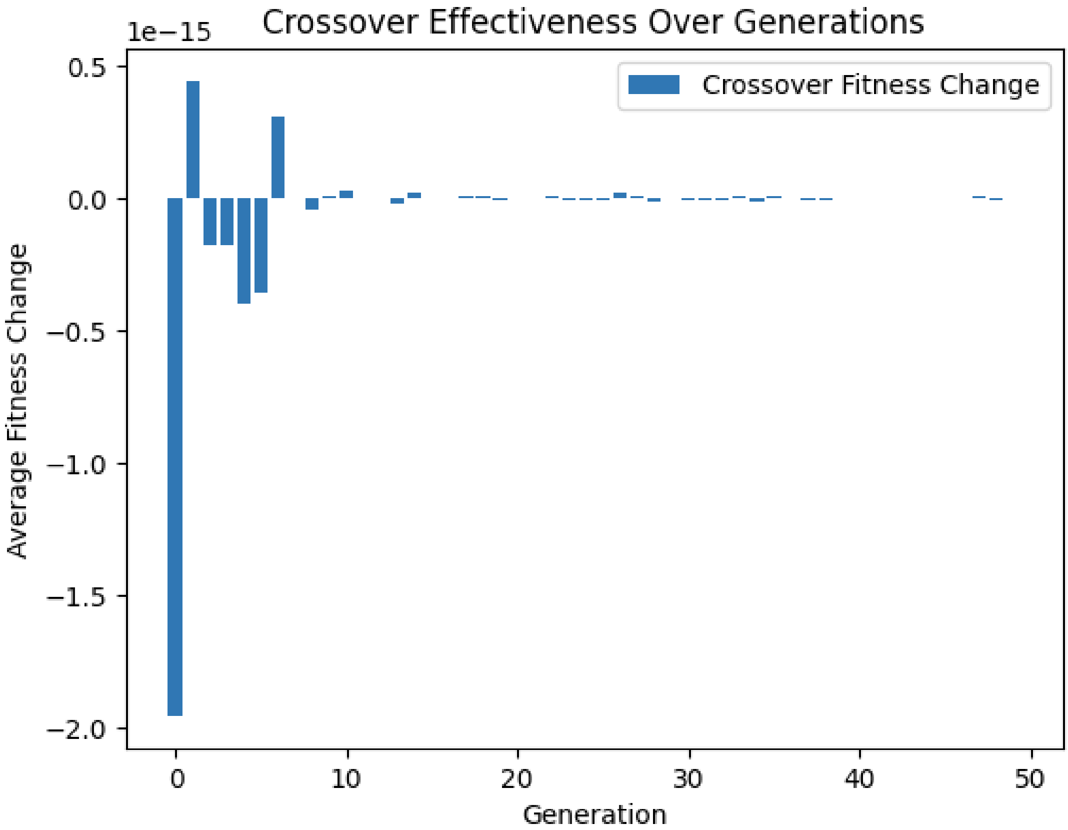
```
# Limit the number of generations to show
num_generations_to_show = 5
limited_genealogy = {k: v for k, v in genealogy.items() if k // population_size <
num_generations_to_show}

G = nx.DiGraph()
for key, values in limited_genealogy.items():
    for value in values:
        G.add_edge(value, key)

plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=300, node_color='skyblue', font_size=6,
font_weight='bold', arrows=True)
plt.title('Genealogy Tracking (Limited to First Few Generations)')
plt.show()

# Interactive Plotting with Plotly
edge_x = []
edge_y = []
for edge in G.edges():
    x0, y0 = pos[edge[0]]
    x1, y1 = pos[edge[1]]
    edge_x.extend([x0, x1, None])
    edge_y.extend([y0, y1, None])

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
node_text = []
for node in G.nodes():
    x, y = pos[node]
    node_x.append(x)
    node_y.append(y)
    node_text.append(str(node))

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers+text',
    text=node_text,
    textposition="bottom center",
```

```
    hoverinfo='text',
    marker=dict(
        showscale=False,
        color='skyblue',
        size=10,
        line_width=2))

fig = go.Figure(data=[edge_trace, node_trace],
            layout=go.Layout(
                title='<br>Interactive Genealogy Tracking',
                titlefont_size=16,
                showlegend=False,
                hovermode='closest',
                margin=dict(b=20,l=5,r=5,t=40),
                annotations=[ dict(
                    text="Interactive plot of the genealogy",
                    showarrow=False,
                    xref="paper", yref="paper") ],
                xaxis=dict(showgrid=False, zeroline=False),
                yaxis=dict(showgrid=False, zeroline=False))
                )
fig.show()
```

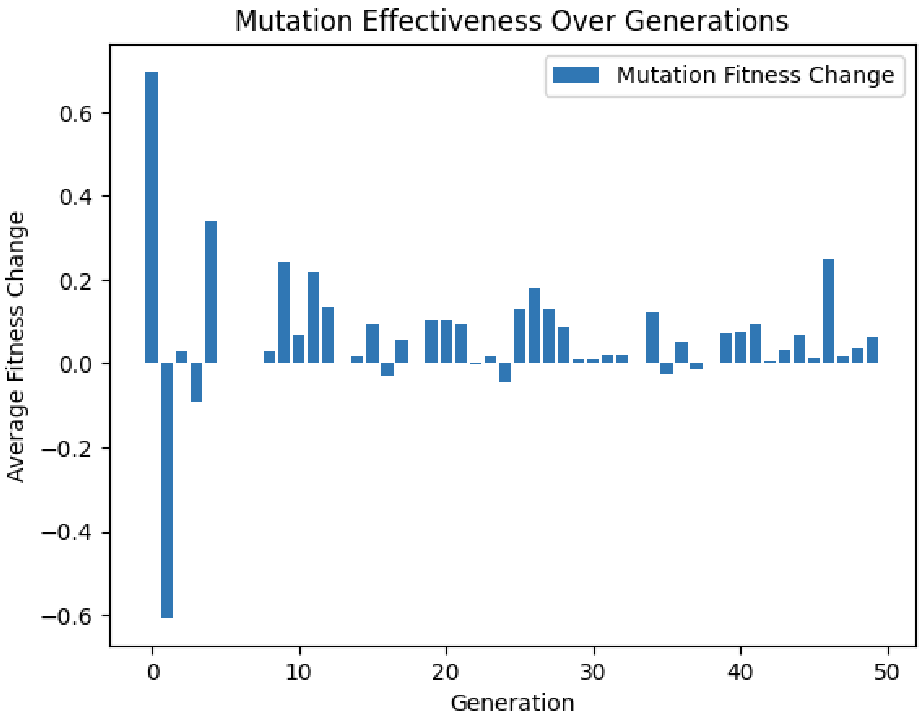


6. מעקב אחר השפעת האופרטורים הגנטיים לאורך הדורות:

```
crossover_fitness_changes = np.array(crossover_fitness_changes)
mutation_fitness_changes = np.array(mutation_fitness_changes)

avg_crossover_fitness_changes = np.mean(crossover_fitness_changes, axis=1)
avg_mutation_fitness_changes = np.mean(mutation_fitness_changes, axis=1)

# Bar chart for crossover effectiveness
plt.figure()
plt.bar(range(num_generations), avg_crossover_fitness_changes, label='Crossover Fitness
Change')
plt.title('Crossover Effectiveness Over Generations')
plt.xlabel('Generation')
plt.ylabel('Average Fitness Change')
plt.legend()
plt.show()

# Bar chart for mutation effectiveness
plt.figure()
plt.bar(range(num_generations), avg_mutation_fitness_changes, label='Mutation Fitness
Change')
plt.title('Mutation Effectiveness Over Generations')
plt.xlabel('Generation')
plt.ylabel('Average Fitness Change')
plt.legend()
plt.show()
```



Crossover Effectiveness Over Generations
1e−15
Crossover Fitness Change
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Average Fitness Change
0
10
20
30
40
50
Generation

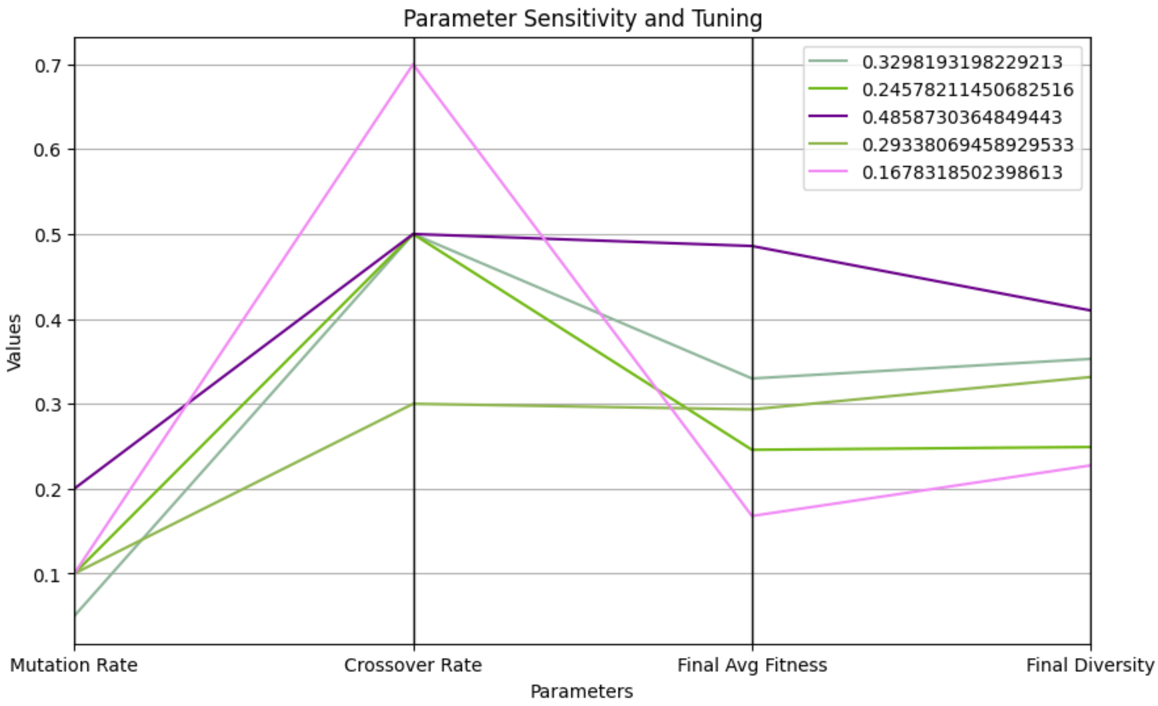


Mutation Effectiveness Over Generations
Mutation Fitness Change
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Average Fitness Change
0
10
20
30
40
50
Generation

7. בדיקת רגישות לפרמטרים שונים של ההרצה:

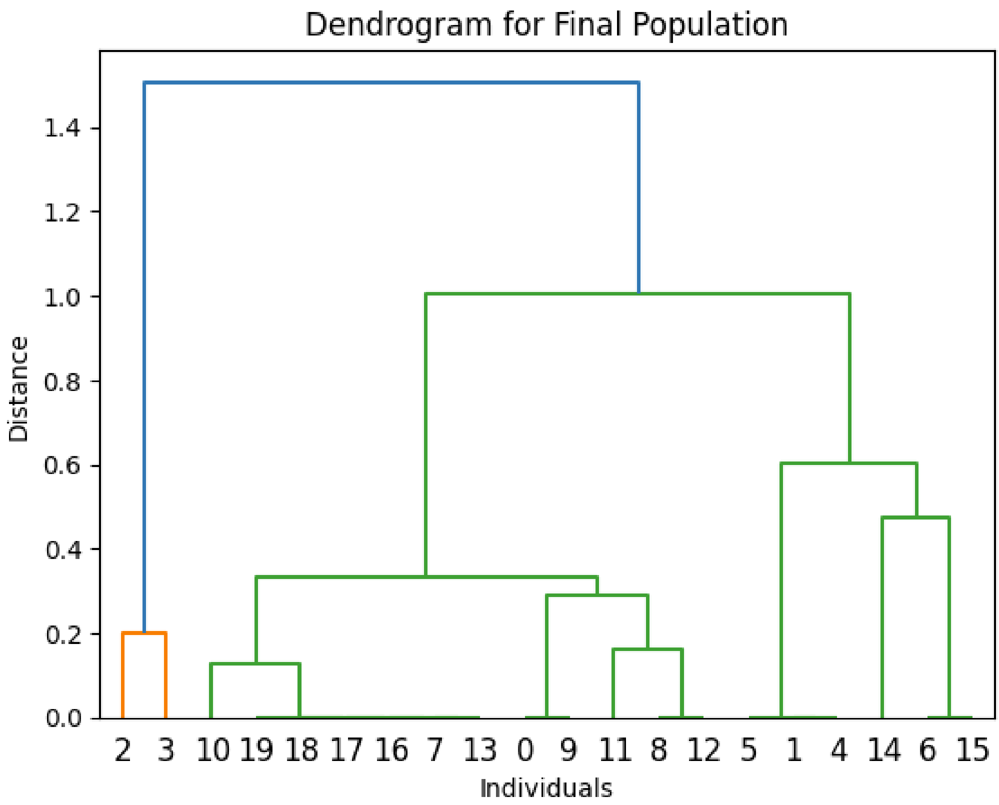
```
# Run GA with different parameter settings
parameter_sets = [
    {'population_size': 20, 'num_generations': 50, 'mutation_rate': 0.05, 'crossover_rate': 0.5},
    {'population_size': 20, 'num_generations': 50, 'mutation_rate': 0.1, 'crossover_rate': 0.5},
    {'population_size': 20, 'num_generations': 50, 'mutation_rate': 0.2, 'crossover_rate': 0.5},
    {'population_size': 20, 'num_generations': 50, 'mutation_rate': 0.1, 'crossover_rate': 0.3},
    {'population_size': 20, 'num_generations': 50, 'mutation_rate': 0.1, 'crossover_rate': 0.7},
]

results = []
for params in parameter_sets:
    avg_fitness_over_time, diversity_over_time, genealogy, crossover_fitness_changes,
mutation_fitness_changes, final_population = run_ga(
        params['population_size'], params['num_generations'], params['mutation_rate'],
params['crossover_rate'])
    final_avg_fitness = avg_fitness_over_time[-1]
    final_diversity = diversity_over_time[-1]
    results.append([params['mutation_rate'], params['crossover_rate'], final_avg_fitness,
final_diversity, final_population])

# Create DataFrame for parallel coordinates plot
results_df = pd.DataFrame(results, columns=['Mutation Rate', 'Crossover Rate', 'Final Avg Fitness',
'Final Diversity', 'Final Population'])

# Parallel Coordinates Plot
plt.figure(figsize=(10, 6))
parallel_coordinates(results_df.drop(columns=['Final Population']), class_column='Final Avg Fitness',
cols=['Mutation Rate', 'Crossover Rate', 'Final Avg Fitness', 'Final Diversity'])
plt.title('Parameter Sensitivity and Tuning')
plt.xlabel('Parameters')
plt.ylabel('Values')
```

plt.show()



8. חלוקה היררכית של האוכלוסיה לצבירים לפי דמיון בין הפרטים באוכלוסייה הסופית

```
# Dendrogram for cluster analysis

Z = linkage(final_population, 'ward')

plt.figure()

dendrogram(Z)

plt.title('Dendrogram for Final Population')

plt.xlabel('Individuals')

plt.ylabel('Distance')

plt.show()
```


Dendrogram for Final Population
Distance
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
2 3 10 19 18 17 16 7 13 0 9 11 8 12 5 1 4 14 6 15
Individuals

למי שממש ב-C++ יש ספריה בשם Qt שבאמצעותה ניתן ליצר ויזואליזציה מקבילה

דוגמא לגרף של ערכים אקראיים לאורך סדרת זמן:

```
#include <QtCharts>
#include <QChartView>
#include <QLineSeries>
#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
   // Create the data series
   QtCharts::QLineSeries *series = new QtCharts::QLineSeries();
   for (int i = 1; i <= 50; ++i) {
      series->append(i, qrand() % 50);  // Random data for illustration
   }
   // Create a chart
   QtCharts::QChart *chart = new QtCharts::QChart();
   chart->legend()->hide();
   chart->addSeries(series);
   chart->createDefaultAxes();
   chart->setTitle("Simple Fitness Chart Example");

   // Create a chart view
   QtCharts::QChartView *chartView = new QtCharts::QChartView(chart);
   chartView->setRenderHint(QPainter::Antialiasing);
   // Show the chart
   chartView->resize(640, 480);
   chartView->show();
   return app.exec();
```

```
}
```