

raport

November 28, 2023

1 Lab 2 - Algorytmy ewolucyjne

```
[13]: from wsilib.algorithms.evo.evo import EvoSolver, StopConditions
      from wsilib.algorithms.evo.individual import TSPIndividualType, DomainIndividualType
      from wsilib.algorithms.evo.genetic_operations import GeneticOperations
      from wsilib.algorithms.evo.selection_methods import SelectionMethods
      from wsilib.algorithms.evo.succession_methods import SuccessionMethods
      from wsilib.utils.function import Function

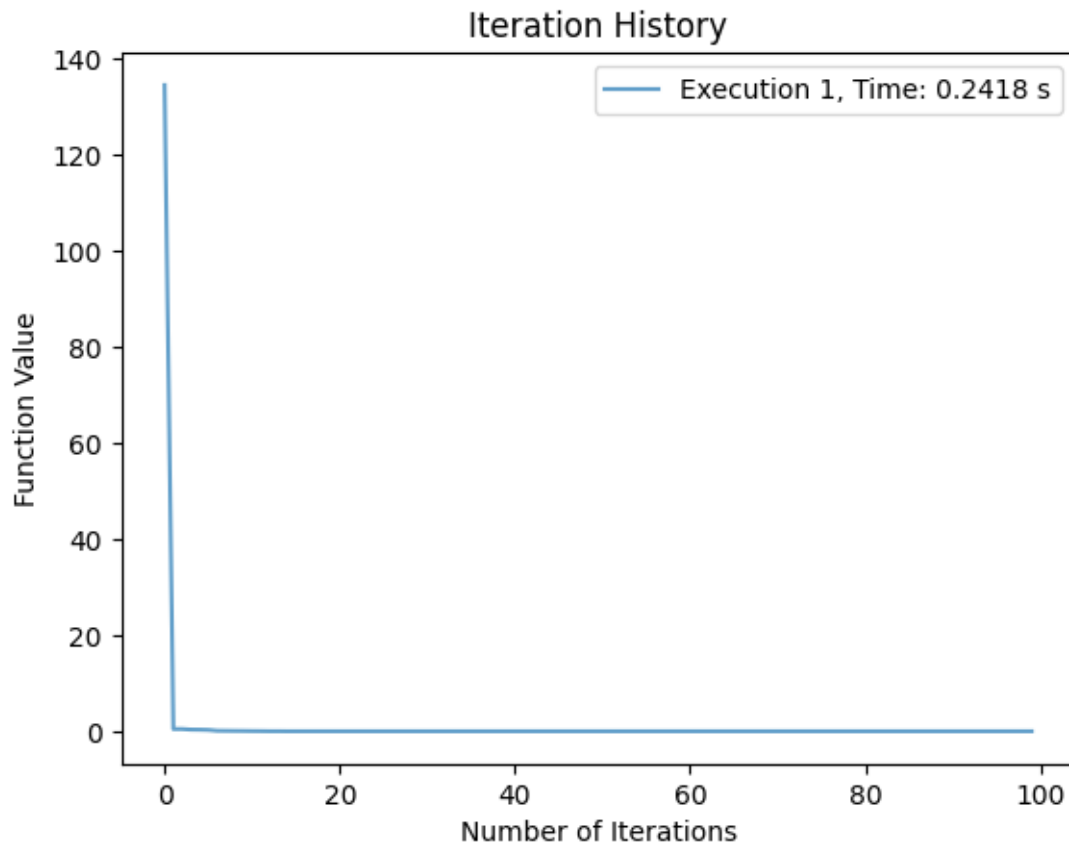
      from src.experiments import generate_cost_function, experiment, params_to_label, avg_f_value, test_sets_generator
      from src.plotting import plot_results, plot_cities
```

1.1 Optymalizacja funkcji kwadratowej

```
[14]: f = Function(
      lambda x: x[0]**2 + x[1]**2,
      dim=2,
      )

      solver = EvoSolver(
          individual_type=DomainIndividualType(20, (-10, 10)),
          population_size=100,
          selection_method=SelectionMethods.tournament_selection(2),
          genetic_operations=[
              GeneticOperations.mutation(0.1),
              GeneticOperations.single_point_crossover(),
          ],
          succession_method=SuccessionMethods.elitism_succession(1),
          stop_conditions=[
              StopConditions.max_iterations(100),
          ]
      )

      res = solver.solve(f)
      plot_results([res], mean=False)
```



1.2 Problem komwojażera

```
[15]: cities = [  
    [35, 51],  
    [113, 213],  
    [82, 280],  
    [322, 340],  
    [256, 352],  
    [160, 24],  
    [322, 145],  
    [12, 349],  
    [282, 20],  
    [241, 8],  
    [398, 153],  
    [182, 305],  
    [153, 257],  
    [275, 190],  
    [242, 75],  
    [19, 229],
```

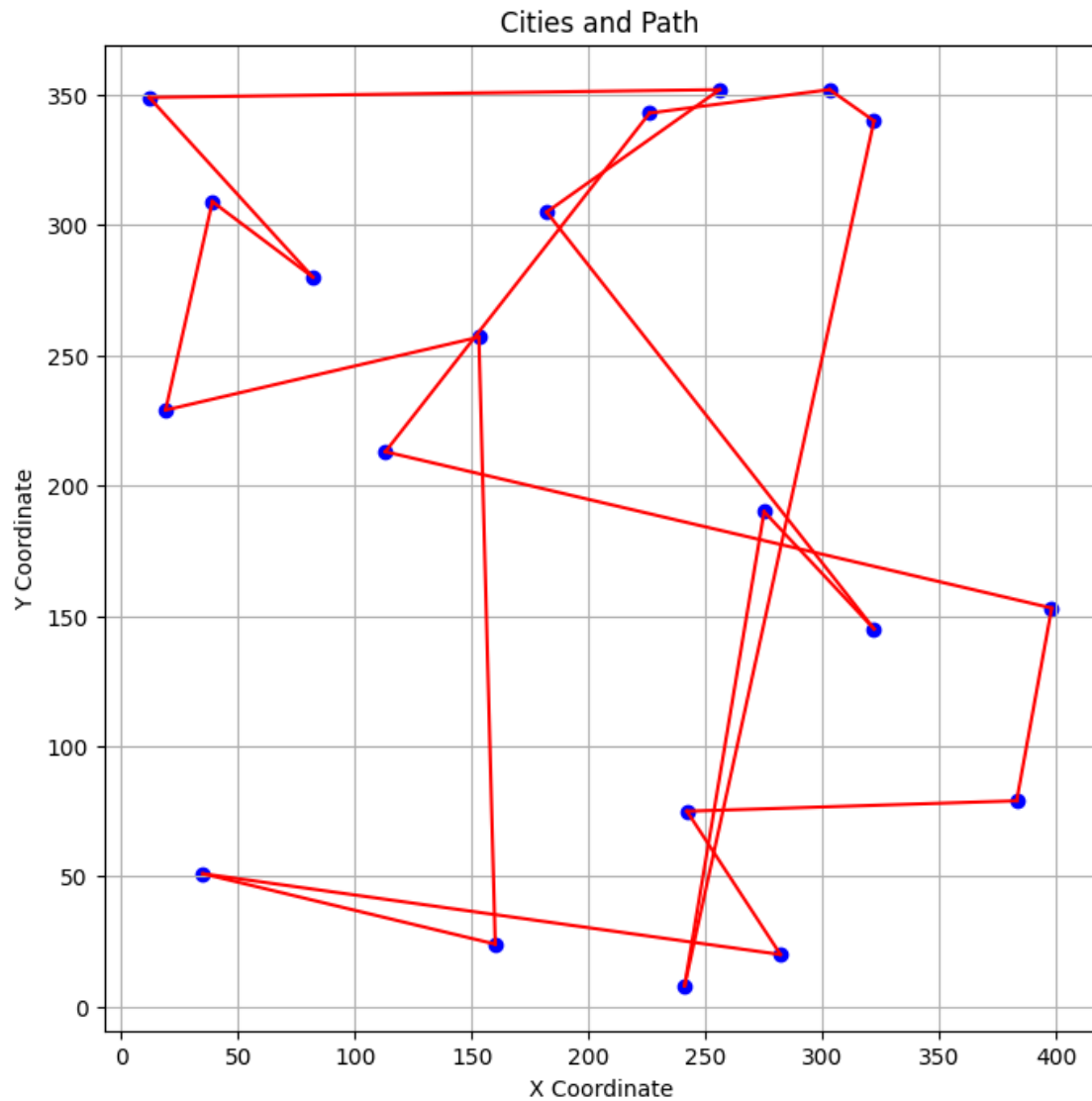
```

[303, 352],
[39, 309],
[383, 79],
[226, 343],
]

solver = EvoSolver(
    individual_type=TSPIndividualType(n_genes=len(cities)),
    population_size=100,
    selection_method=SelectionMethods.tournament_selection(2),
    genetic_operations=[
        GeneticOperations.mutation(0.1),
        GeneticOperations.tsp_crossover(0.5),
    ],
    succession_method=SuccessionMethods.generational_succession(),
    stop_conditions=[
        StopConditions.max_iterations(100),
    ],
)

result = solver.solve(generate_cost_function(cities))
plot_cities(cities, result.x)

```



1.3 Eksperyment: szukanie optymalnych hiperparametrów dla problemu kom-wojażera

```
[16]: exp = experiment(params={
    'population_size': [10, 100],
    'selection_method': [SelectionMethods.tournament_selection(2),
↪ SelectionMethods.tournament_selection(5)],
    'genetic_operations': [
        [
            GeneticOperations.mutation()
        ],
        [
```

```

        GeneticOperations.mutation(),
        GeneticOperations.tsp_crossover(0.5),
    ],
    [
        GeneticOperations.mutation(),
        GeneticOperations.tsp_crossover(0.7),
    ]
],
'succession_method': [SuccessionMethods.generational_succession(),
↳ SuccessionMethods.elitism_succession(1)],
'stop_conditions': [
    StopConditions.max_iterations(1000),
],
}, n_sets=10)

best = (None, None, None, None)
for params, results, progress in exp:
    label = params_to_label(params)
    print(progress)
    print(label)
    plot_results(results)
    if(best == (None, None, None, None)):
        best = (avg_f_value(results), label, results, params)
    else:
        if(avg_f_value(results) < best[0]):
            best = (avg_f_value(results), label, results, params)

print("Best config:")
print(best[1])
plot_results(best[2])
print("Średnia wartość funkcji kosztu: " + str(best[0]))
best_params = best[3]

```

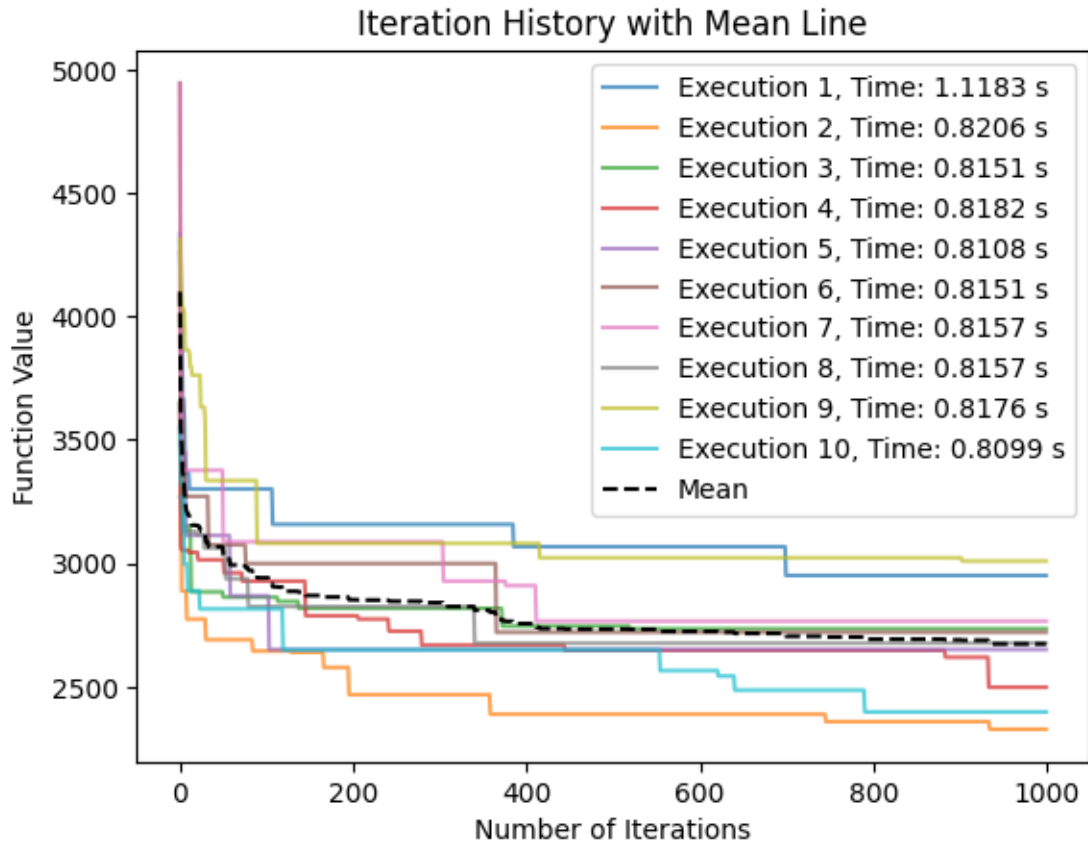
1/24

Params:

```

population_size: 10
selection_method: tournament_selection(2)
genetic_operations:
    mutation
succession_method: generational_succession
stop_conditions:
    max_iterations(1000)

```



```

KeyboardInterrupt                                Traceback (most recent call last)
/Users/aszokalski/Developer/studia/Sem 3/WSI/notebooks/lab2/lab2.ipynb Cell 8
↳ line 2
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=0'>1</a> exp =
↳ experiment(params={
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=1'>2</a>
↳ 'population_size': [10, 100],
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=2'>3</a>
↳ 'selection_method': [SelectionMethods.tournament_selection(2),
↳ SelectionMethods.tournament_selection(5)],
    (...)
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=19'>20</a> ],
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=20'>21</a> }, n_sets=10)
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳ WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=22'>23</a> best = (None,
↳ None, None, None)

```

```

---> <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=23'>24</a> for params,
↳results, progress in exp:
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=24'>25</a>     label =
↳params_to_label(params)
    <a href='vscode-notebook-cell:/Users/aszokalski/Developer/studia/Sem%203/
↳WSI/notebooks/lab2/lab2.ipynb#X10sZmlsZQ%3D%3D?line=25'>26</a>     ↵
↳print(progress)

```

File ~/Developer/studia/Sem 3/WSI/notebooks/lab2/experiments.py:125, in ↵

```

↳experiment(params, n_sets, log)
    123 for cities in test_sets:
    124     loss_function = generate_cost_function(cities)
--> 125     result = solver.solve(loss_function, log=log)
    126     results.append(result)
    128 yield params, results, f"{i + 1}/{total}"

```

File ~/Developer/studia/Sem 3/WSI/.venv/lib/python3.10/site-packages/wsilib/

```

↳algorithms/evo/evo.py:122, in EvoSolver.solve(self, problem, x0, log,
↳log_interval_time)
    109     return EvoResult(
    110         x0=x0,
    111         x=iteration.x,
    (...)
    117         history=history,
    118     )
    120 history.append(iteration)
--> 122 new_population = self.selection_method.function(population, problem)
    124 mutated_population = new_population
    125 for operation in self.genetic_operations:

```

File ~/Developer/studia/Sem 3/WSI/.venv/lib/python3.10/site-packages/wsilib/

```

↳algorithms/evo/selection_methods.py:25, in SelectionMethods.
↳tournament_selection.<locals>.selection_method(population, cost_function)
    19 def selection_method(
    20     population: Population, cost_function: Function
    21 ) -> Population:
    22     """Select k individuals from population using tournament selection.
↳"""
    24     return np.array(
--> 25         [
    26             min(
    27                 [
    28                     population[i]
    29                     for i in np.random.choice(
    30                         len(population), size=k, replace=False
    31                     )
    32             ],

```

```

33         key=cost_function.f,
34     )
35     for _ in range(len(population))
36 ]
37 )

```

File ~/Developer/studia/Sem 3/WSI/.venv/lib/python3.10/site-packages/wsilib/
↳ algorithms/evo/selection_methods.py:29, in <listcomp>(.0)

```

19 def selection_method(
20     population: Population, cost_function: Function
21 ) -> Population:
22     """Select k individuals from population using tournament selection.
↳ """
23
24     return np.array(
25         [
26             min(
27                 [
28                     population[i]
---> 29                     for i in np.random.choice(
30                         len(population), size=k, replace=False
31                     )
32                 ],
33                 key=cost_function.f,
34             )
35             for _ in range(len(population))
36         ]
37     )

```

File numpy/random/mtrand.pyx:981, in numpy.random.mtrand.RandomState.choice()

File ~/Developer/studia/Sem 3/WSI/.venv/lib/python3.10/site-packages/numpy/core
↳ fromnumeric.py:3100, in prod(a, axis, dtype, out, keepdims, initial, where)

```

2979 @array_function_dispatch(_prod_dispatcher)
2980 def prod(a, axis=None, dtype=None, out=None, keepdims=np._NoValue,
2981         initial=np._NoValue, where=np._NoValue):
2982     """
2983     Return the product of array elements over a given axis.
2984
2985     (...)
3098     10
3099     """
-> 3100     return _wrapreduction(a, np.multiply, 'prod', axis, dtype, out,
3101                           keepdims=keepdims, initial=initial,
↳ where=where)

```

File ~/Developer/studia/Sem 3/WSI/.venv/lib/python3.10/site-packages/numpy/core
↳ fromnumeric.py:88, in _wrapreduction(obj, ufunc, method, axis, dtype, out,
↳ **kwargs)

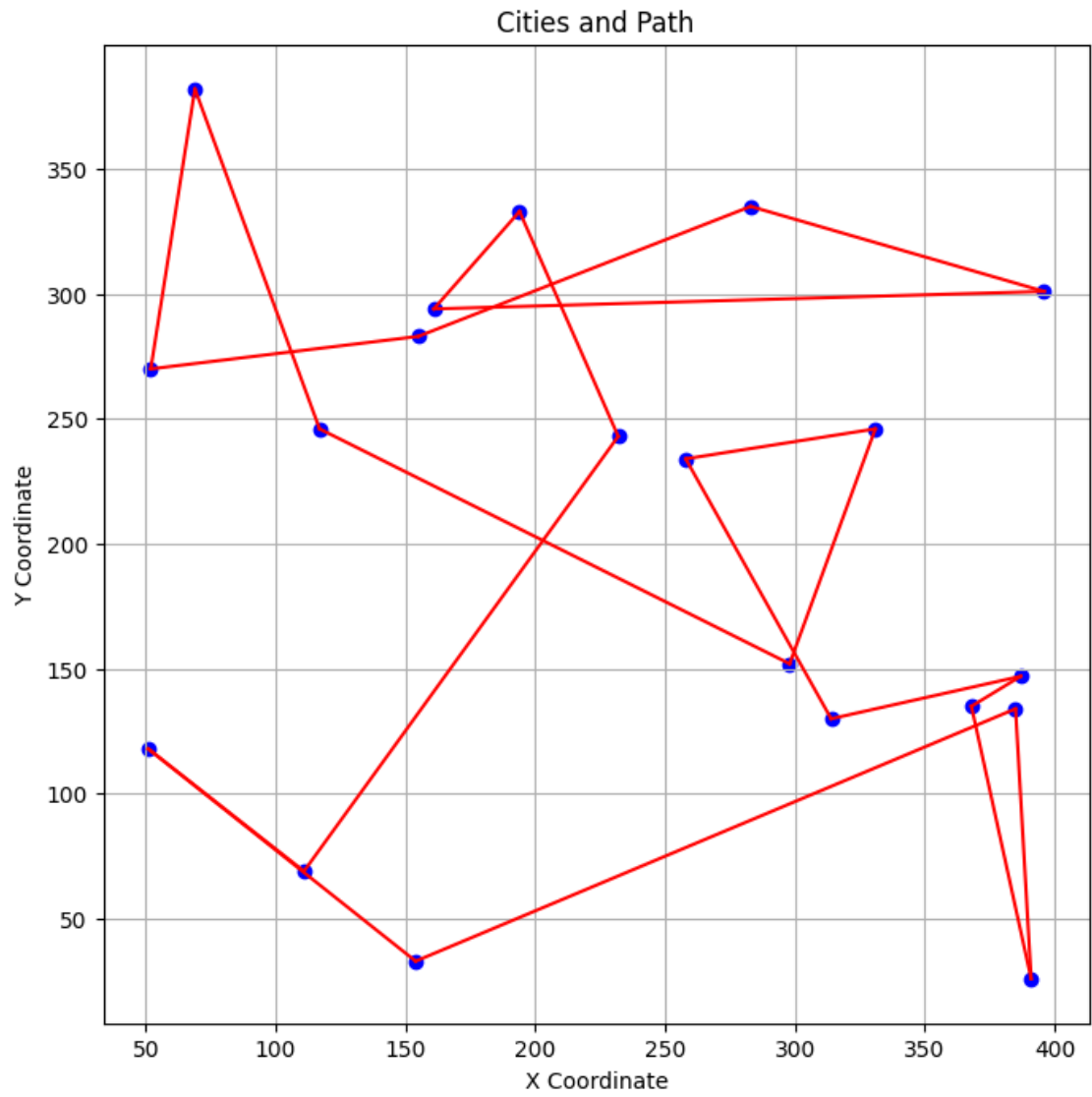

```
85         else:
86             return reduction(axis=axis, out=out, **passkwargs)
---> 88 return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

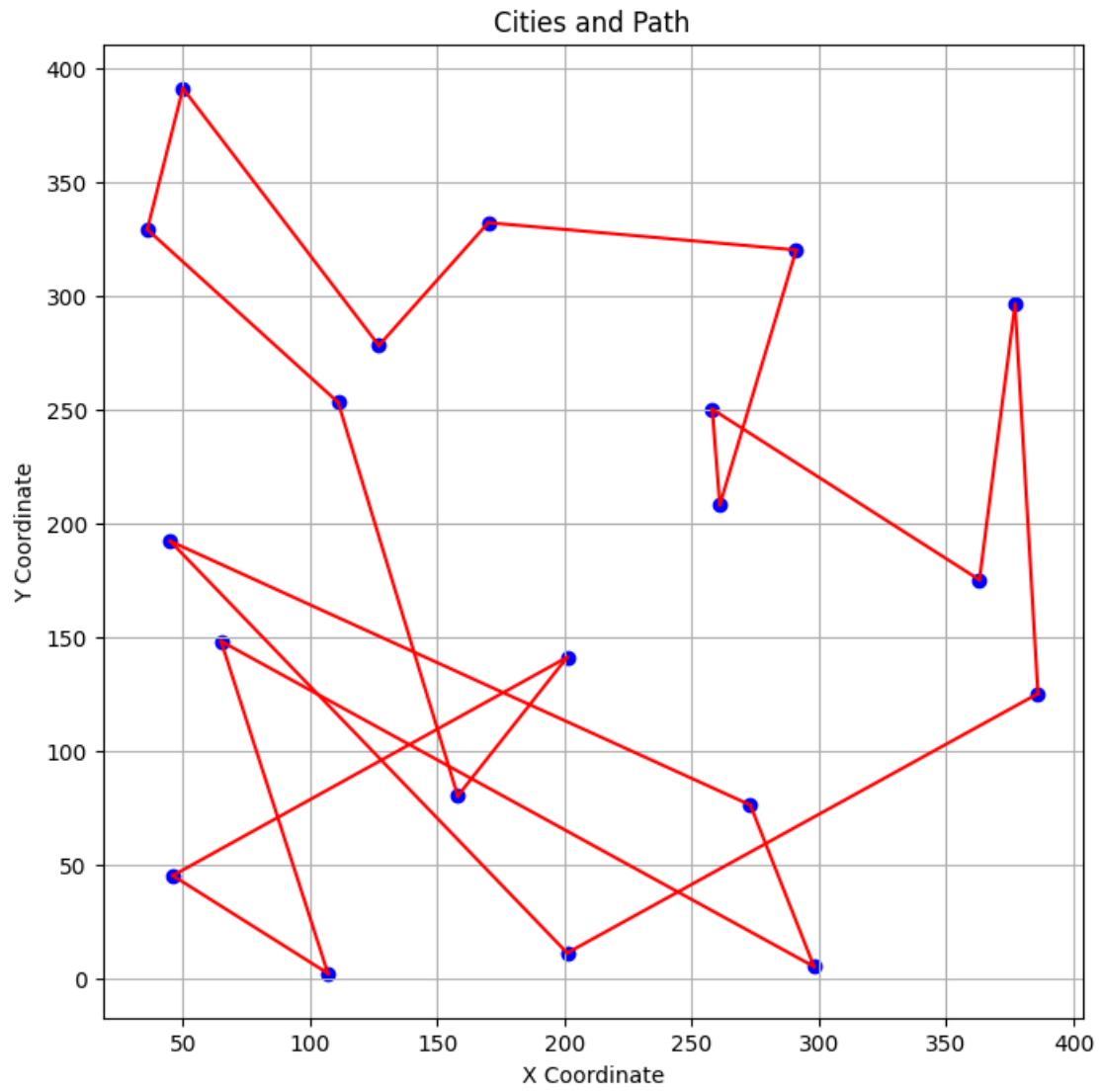
KeyboardInterrupt:

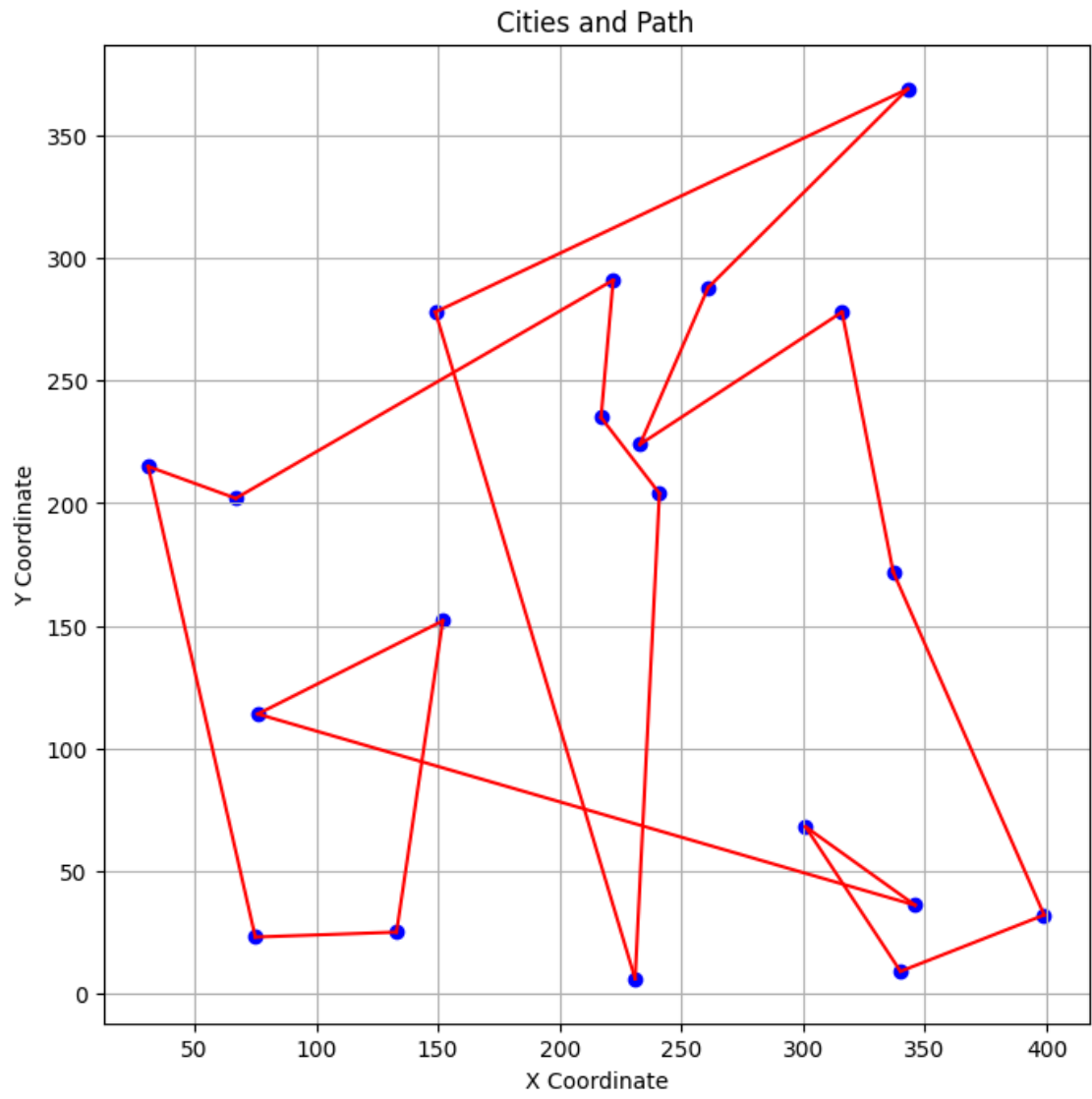
1.4 Problem komwojażera na zoptymizowanym algorytmie ewolucyjnym

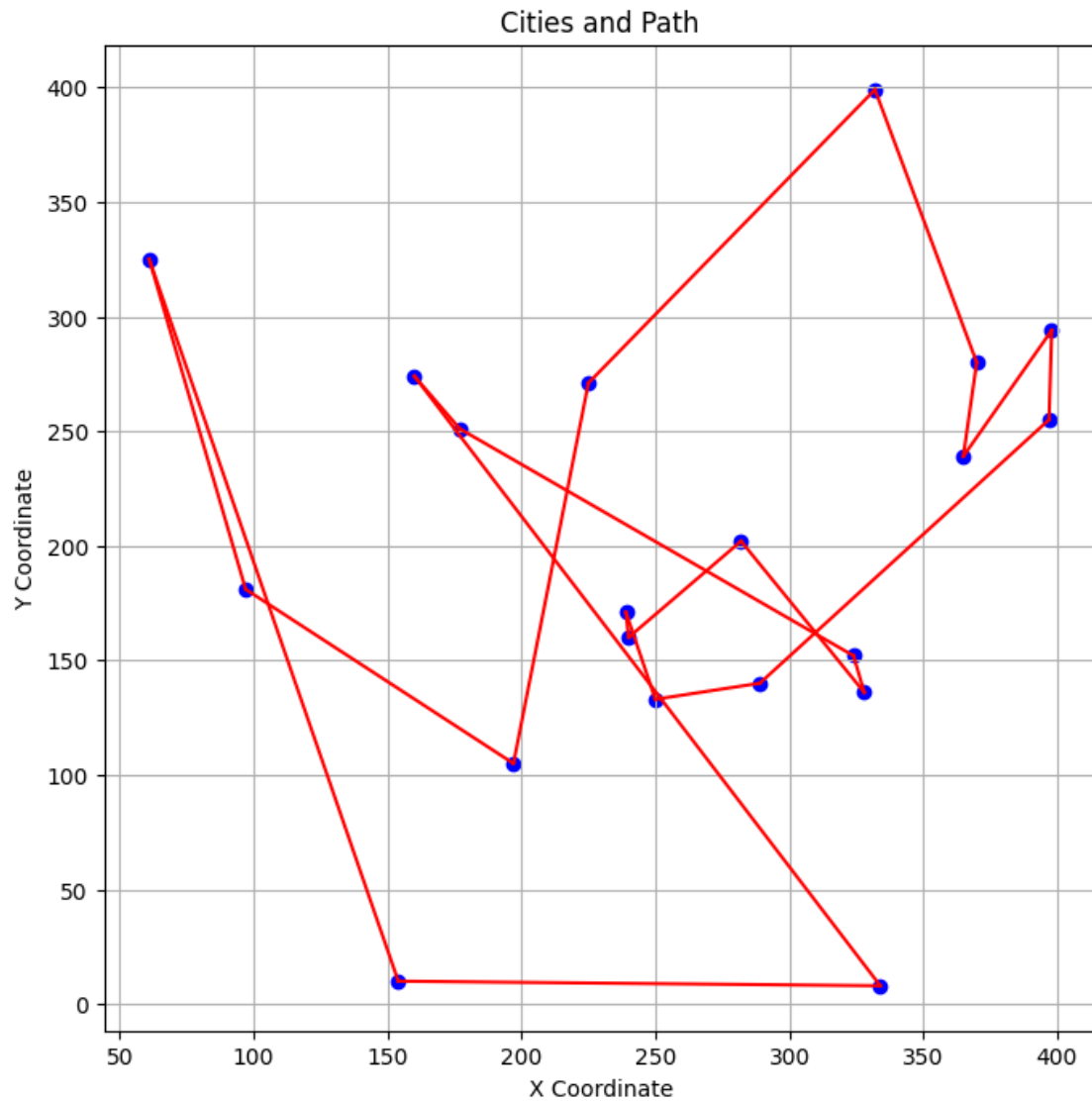
```
[ ]: solver_opt = EvoSolver(
    individual_type=TSPIndividualType(n_genes=len(cities)),
    population_size=best_params['population_size'],
    selection_method=best_params['selection_method'],
    genetic_operations=best_params['genetic_operations'],
    succession_method=best_params['succession_method'],
    stop_conditions=best_params['stop_conditions'],
)

for cities2 in test_sets_generator(4):
    result = solver_opt.solve(generate_cost_function(cities2))
    plot_cities(cities2, result.x)
```









1.5 Wnioski

- Większy rozmiar populacji daje lepsze wyniki - lepsza zdolność eksploracyjna algorytmu
- Selekcja turniejowa dla par daje podobne wyniki co dla grup 5 elementowych.
- Zastosowanie mutacji oraz krzyżowania daje najlepsze wyniki.
- Krzyżowanie najlepiej wykonywać z parametrem $\alpha=0.5$ czyli dzielić osobnika na dwie równe części
- Lepszą metodą sukcesji okazała się być sukcesja generacyjna - prawdopodobnie zwiększa to zdolność eksploracyjną algorytmu