Lab 2 - Algorytmy ewolucyjne

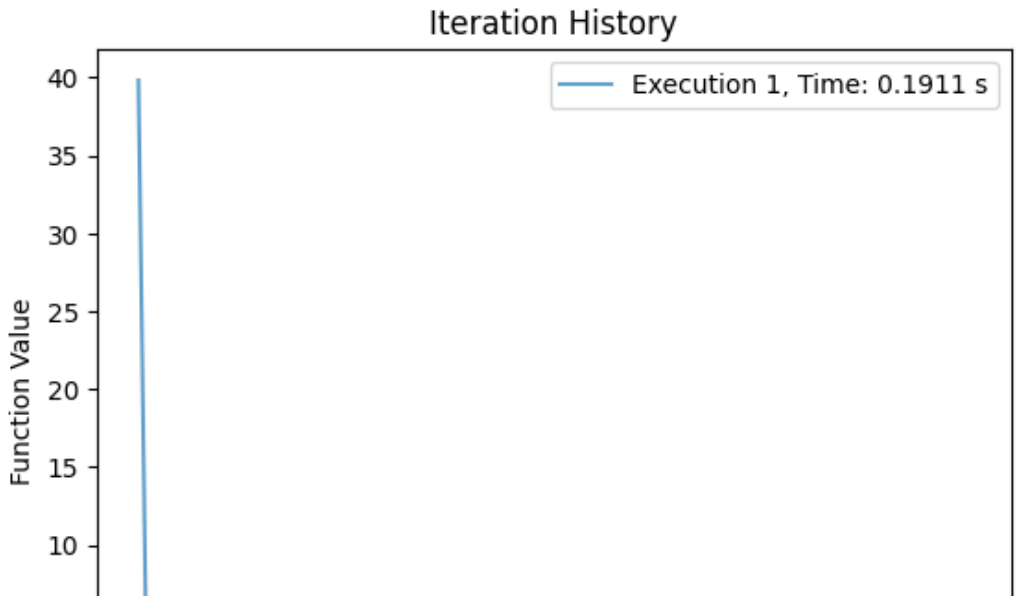
In [5]:

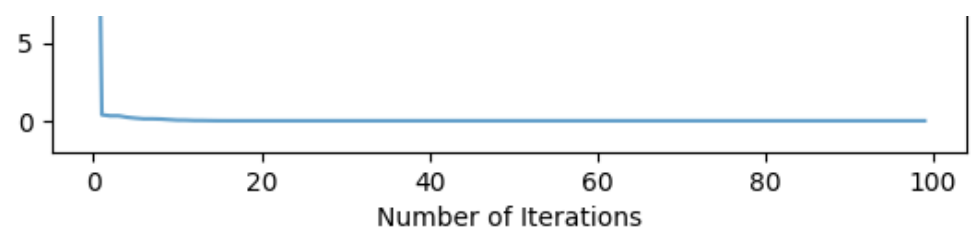
```
# all explicilty imported modules
from imports import (
    TSPIndividualType,
    DomainIndividualType,
    GeneticOperations,
    SelectionMethods,
    SuccessionMethods,
    EvoSolver,
    StopConditions,
    Function
)

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

Optymalizacja funkcji kwadratowej

In [6]:

```
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)
```

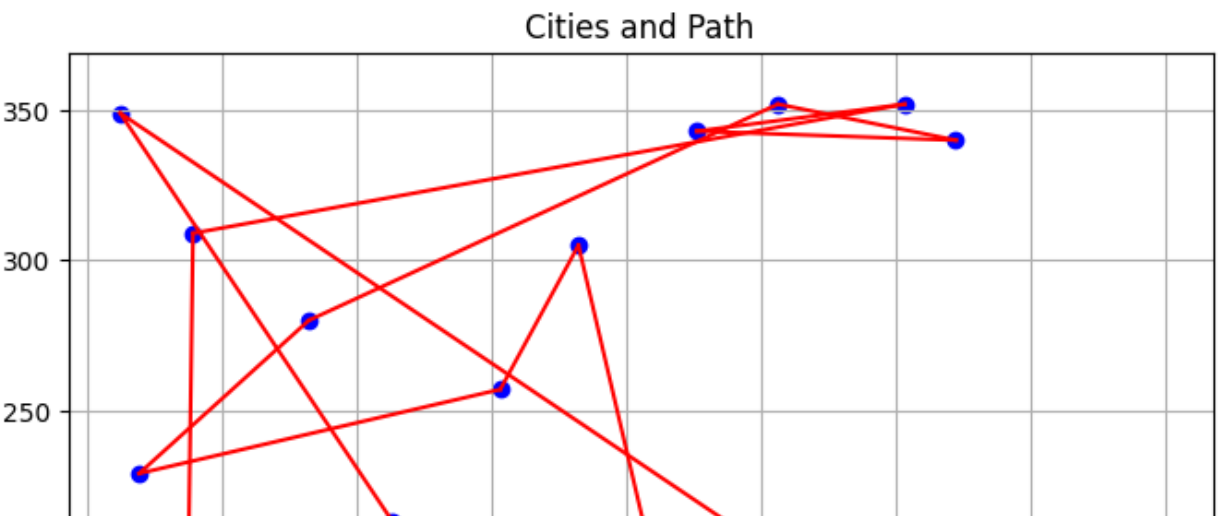


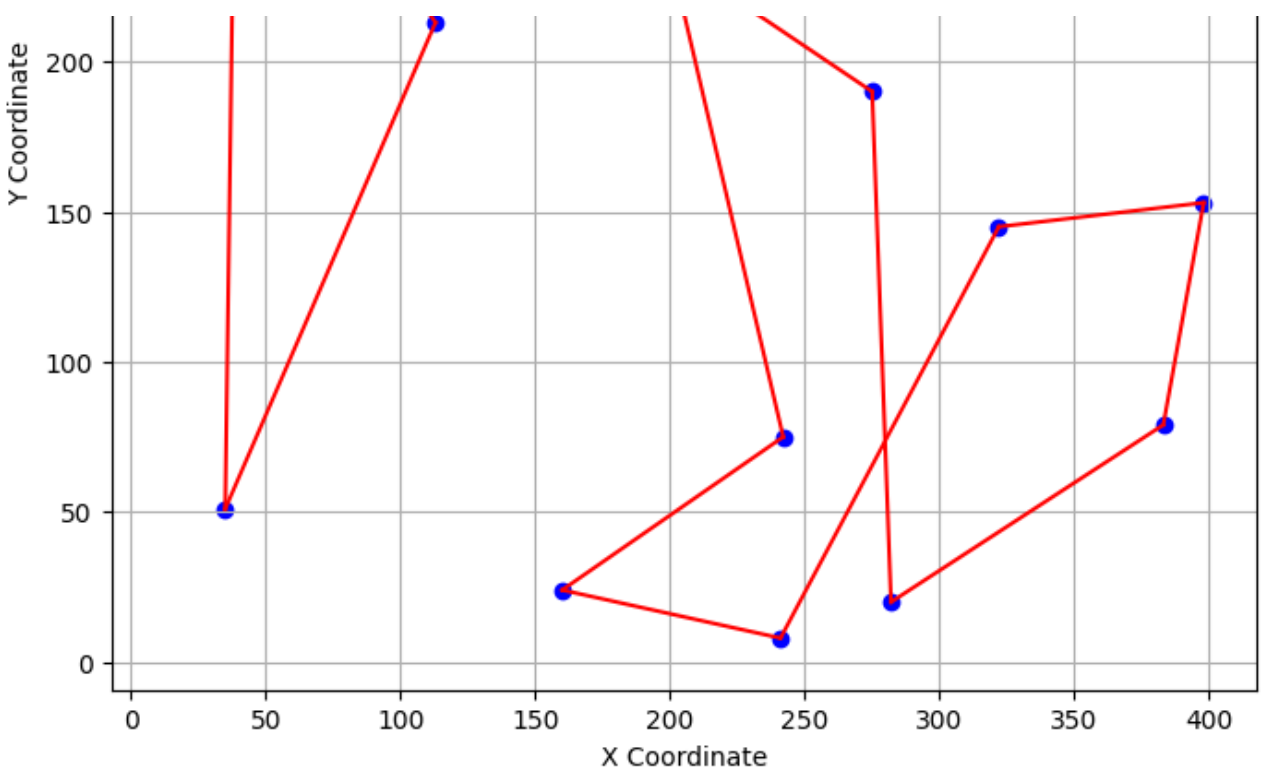


Problem komwojażera

In [7]:

```
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)
```





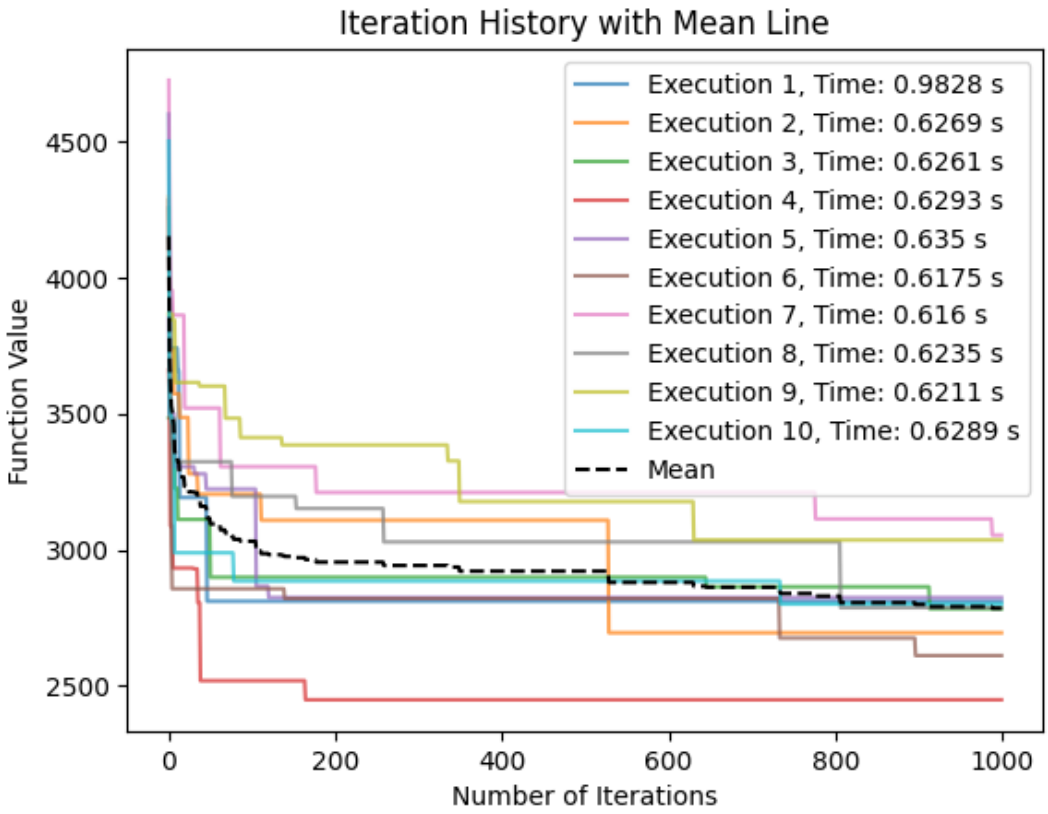
Eksperyment: szukanie optymalnych hiperparametrów dla problemu komwojażera

In [8]:

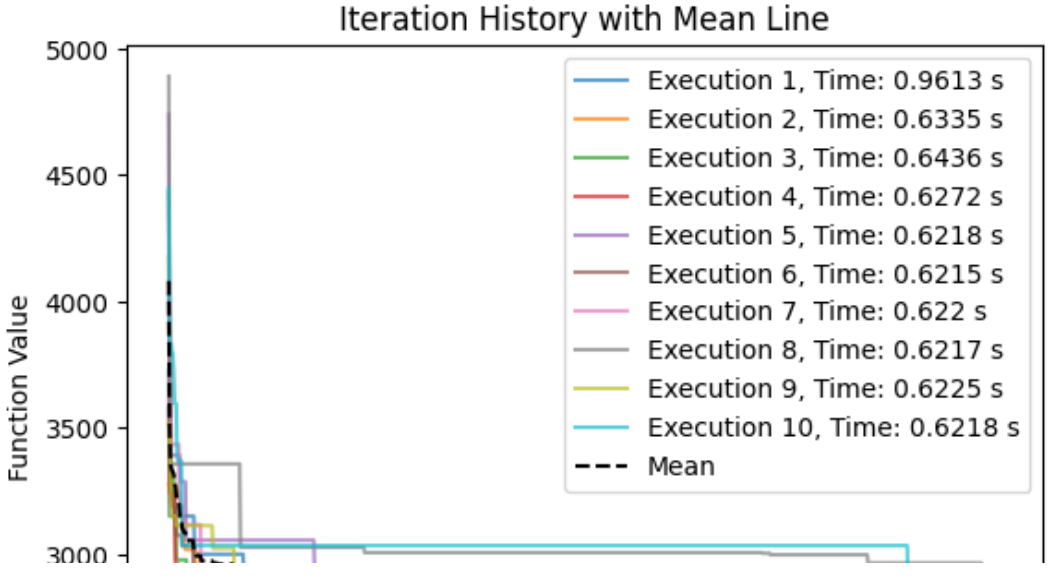
```
exp = experiment(params={
    'population_size': [10, 100],
    'selection_method': [SelectionMethods.tournament_selection(2), SelectionMethods.tour
nament_selection(5)],
    'genetic_operations': [
        [
            GeneticOperations.mutation()
        ],
        [
            GeneticOperations.mutation(),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

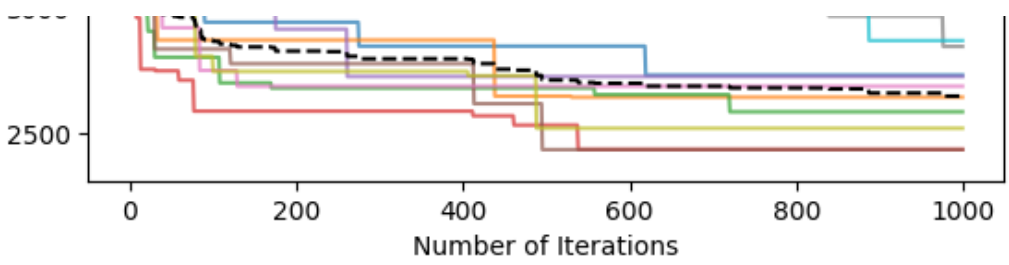
```
plot_results(best[2])
print("Średnia wartość funkcji kosztu: " + str(best[0]))
best_params = best[3]
```

```
1/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

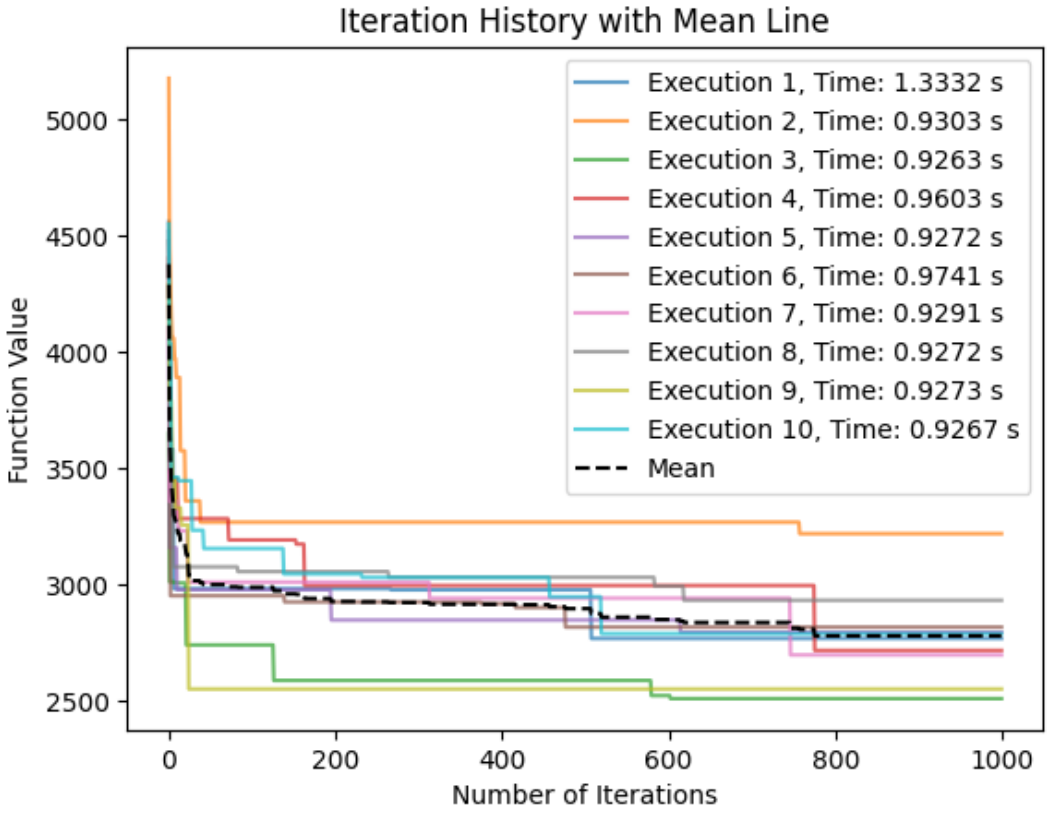



```
2/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```

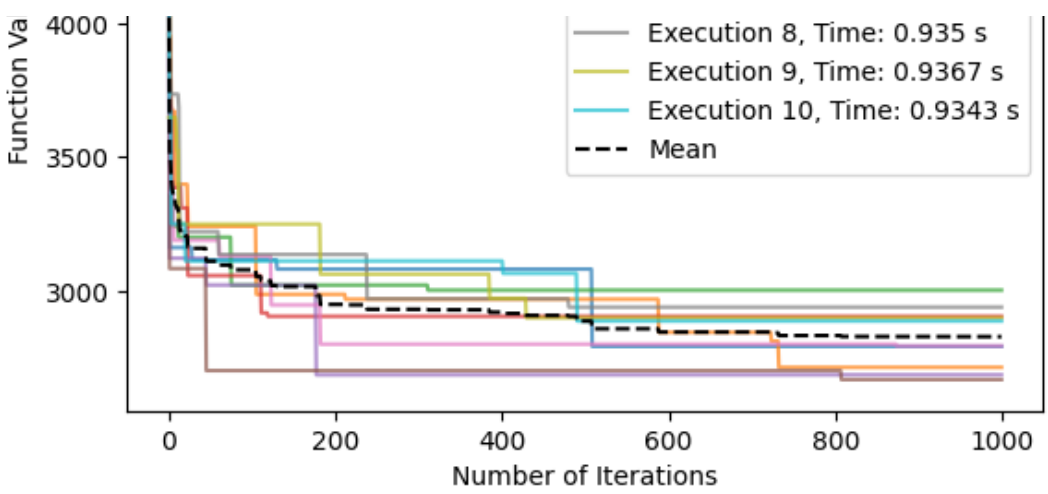




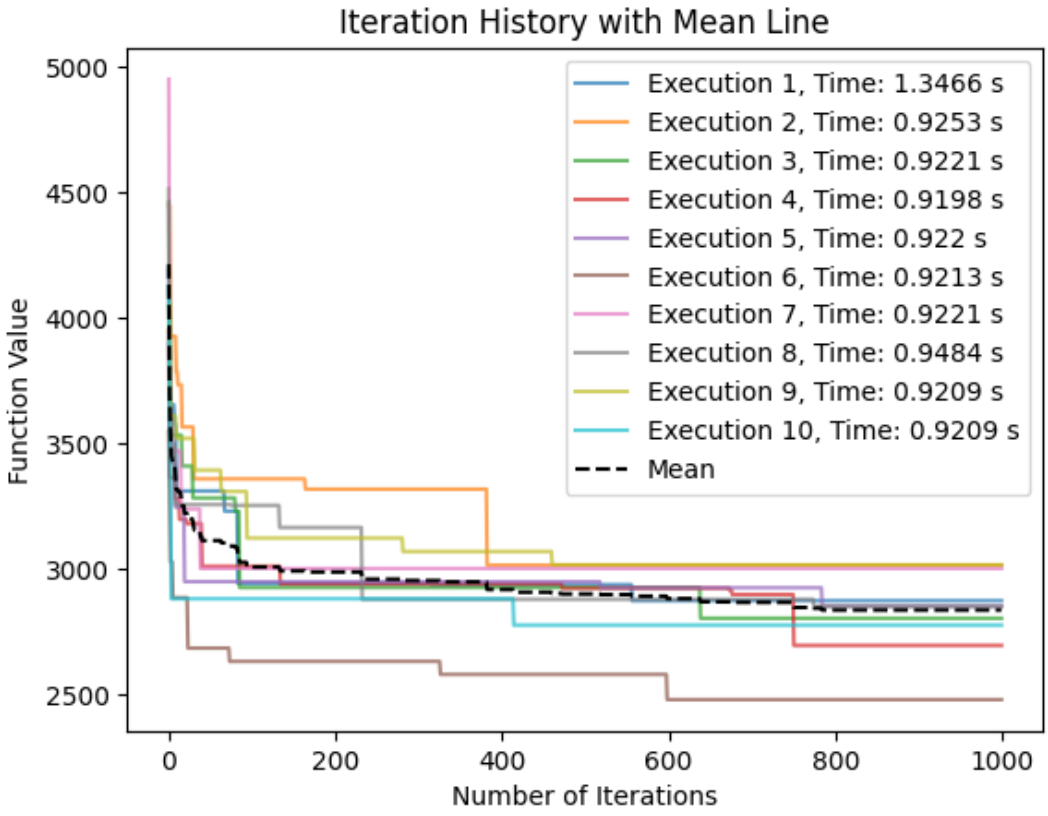

```
3/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```



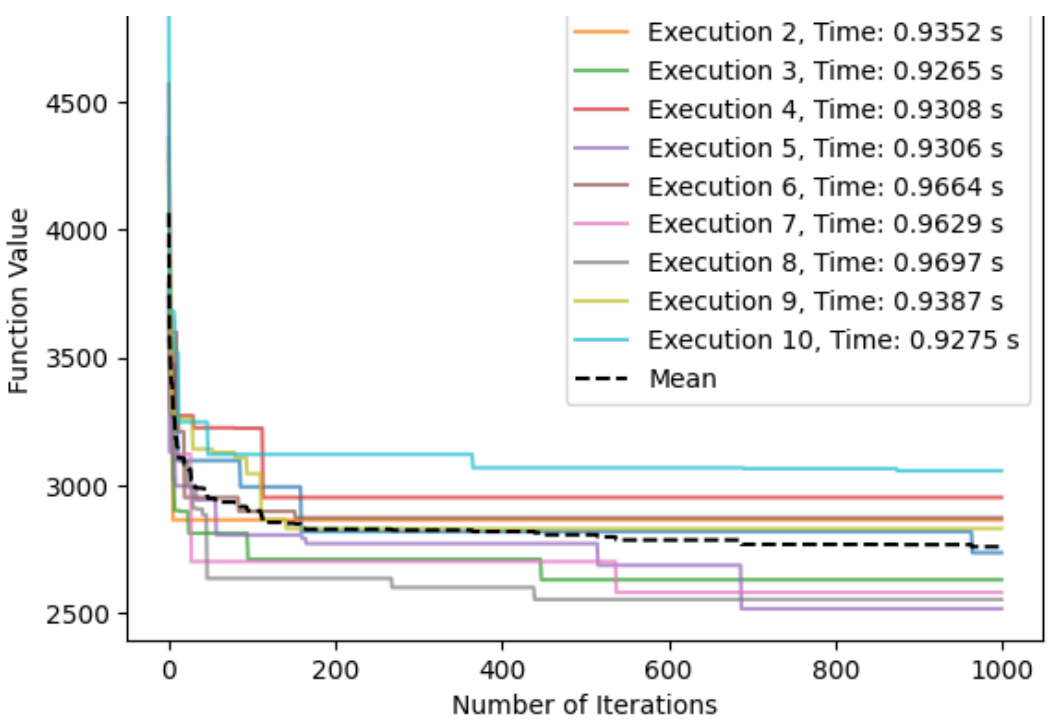

```
4/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```




```
5/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

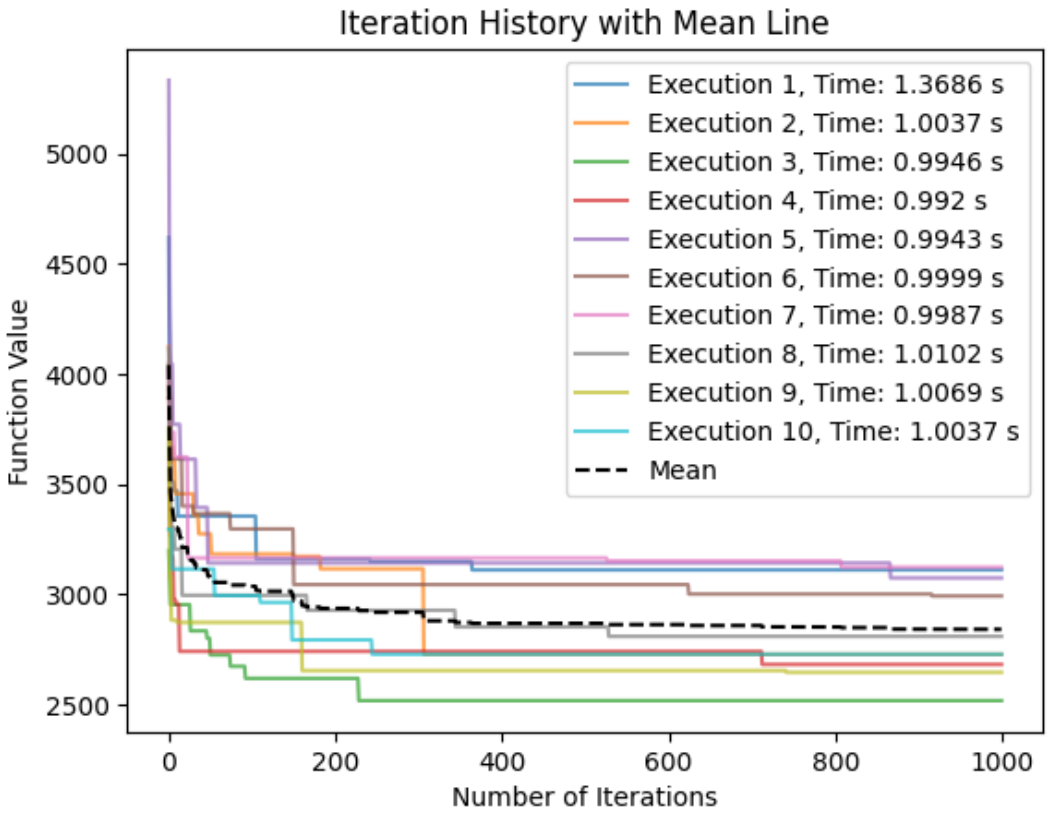



```
6/24
Params:
    population_size: 10
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```



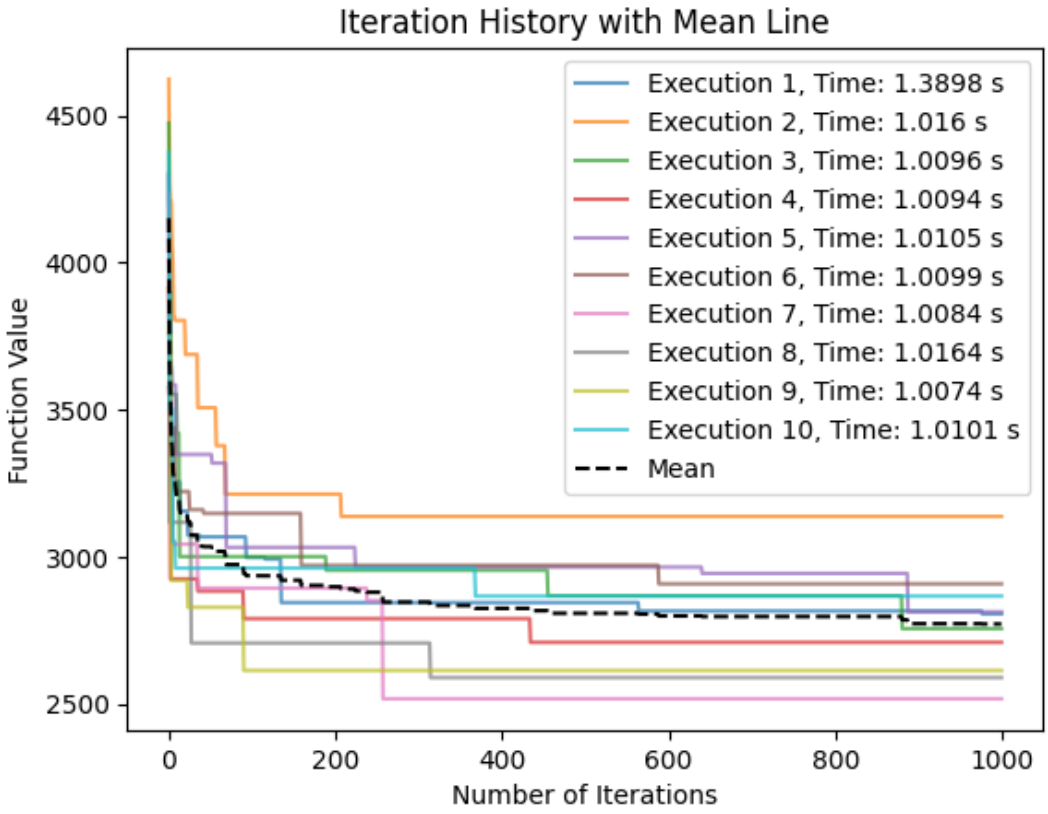
7/24

```
Params:
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

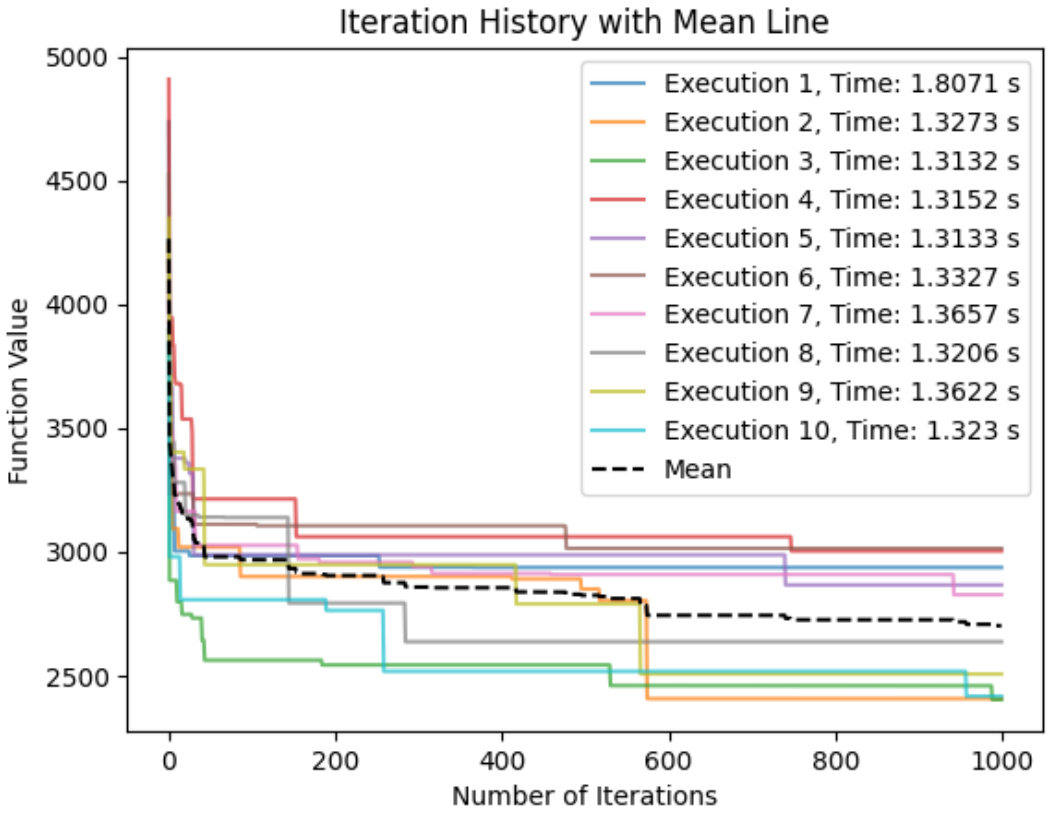


8/24

```
Params:
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```

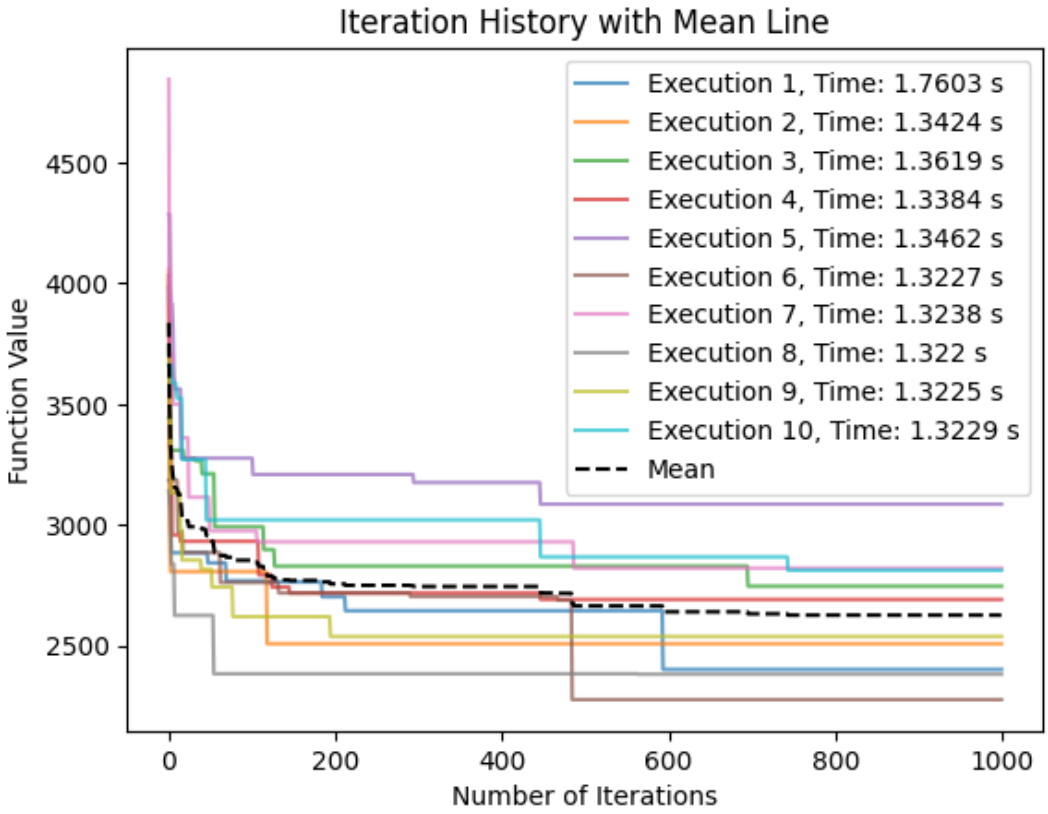


9/24
Params:
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)



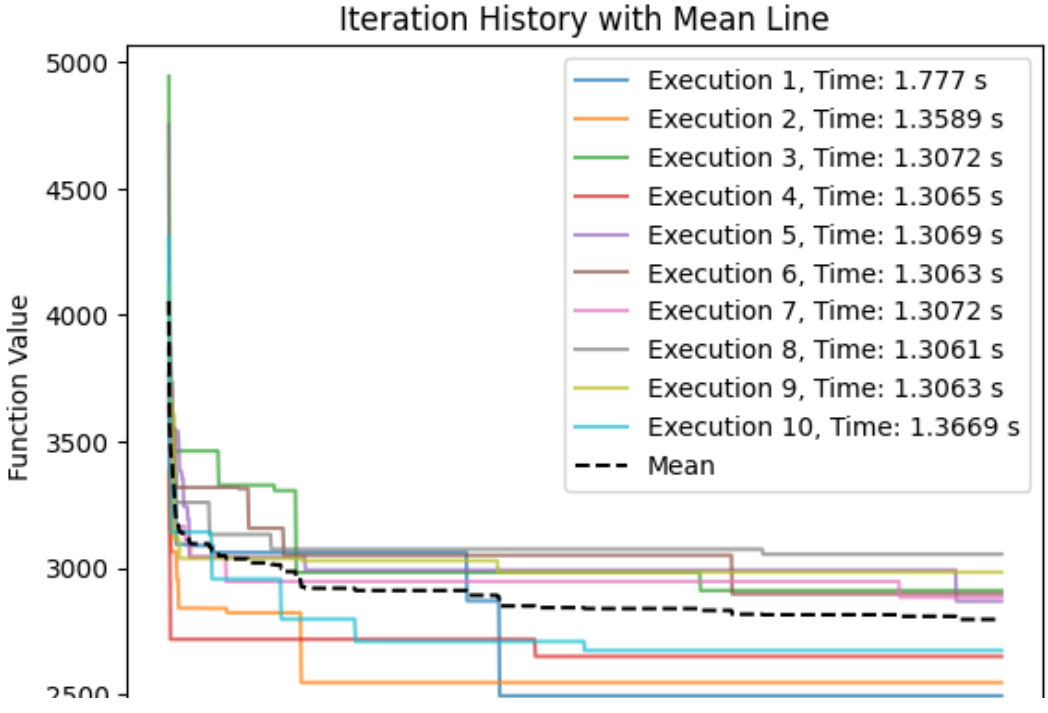
10/24
Params:
    population_size: 10

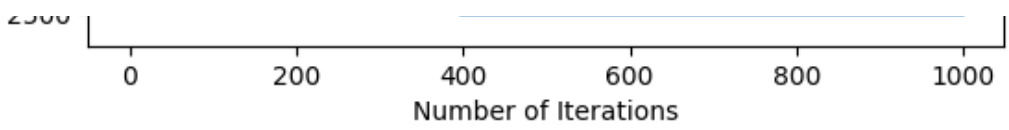
```
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```



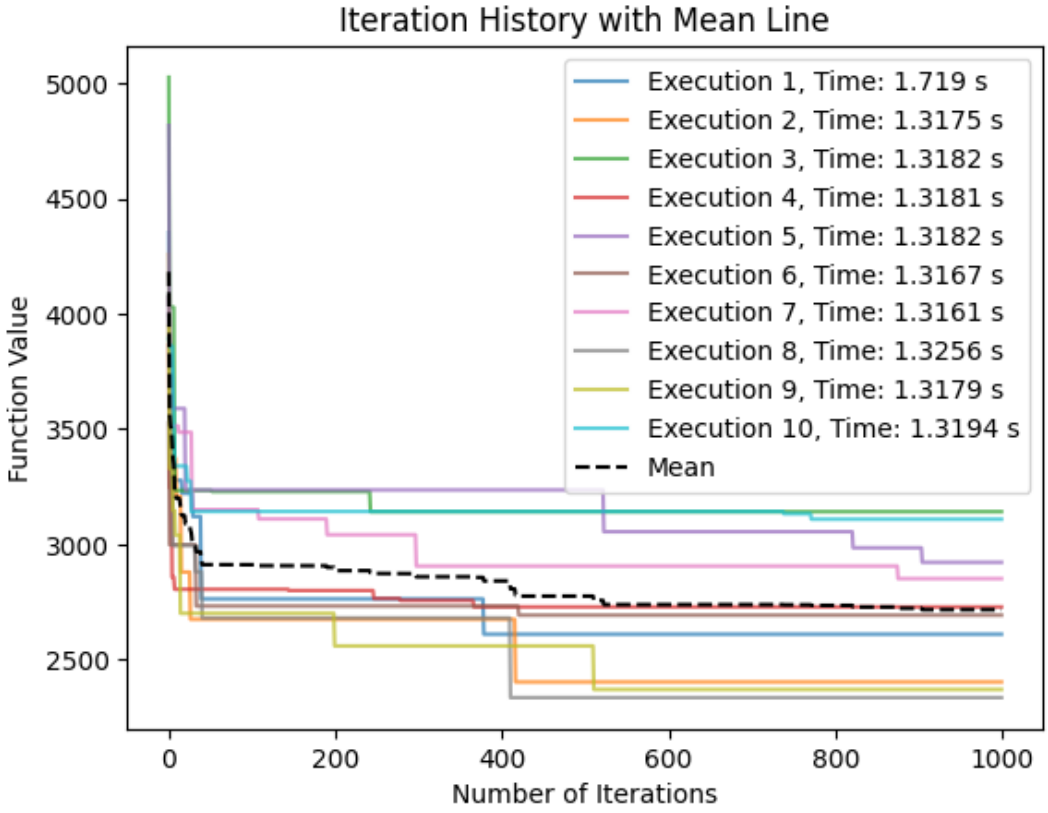
11/24

```
Params:
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

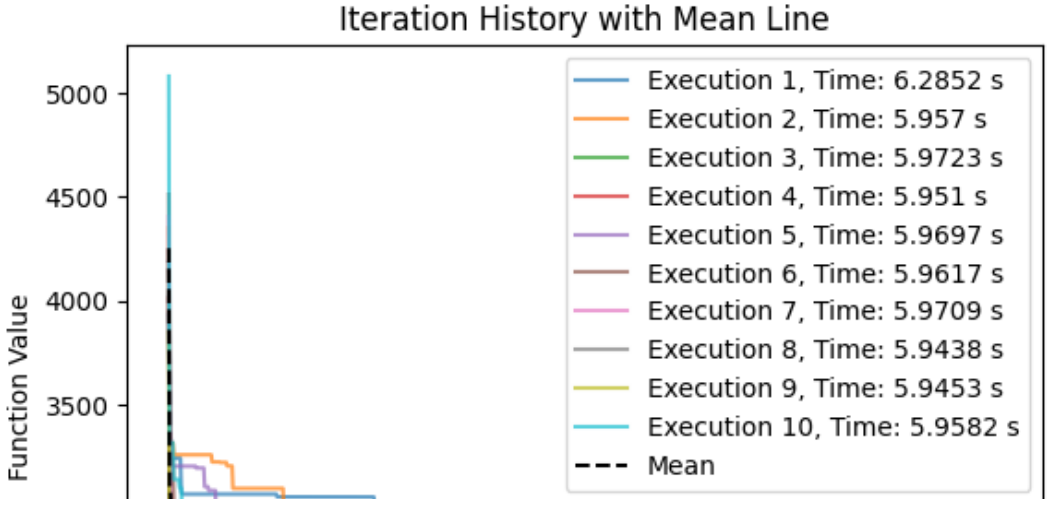


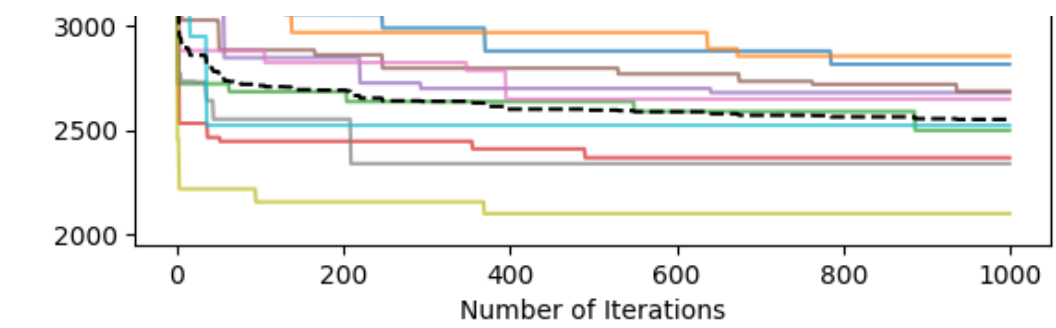



```
12/24
Params:
    population_size: 10
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```

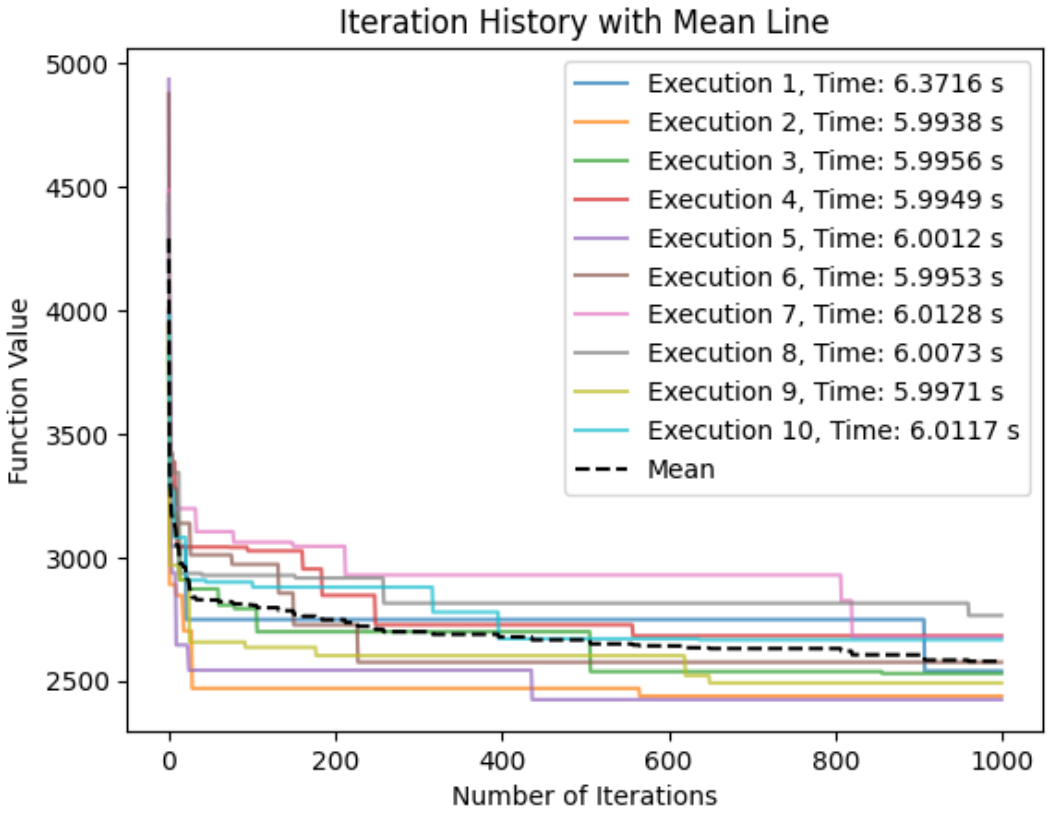



```
13/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

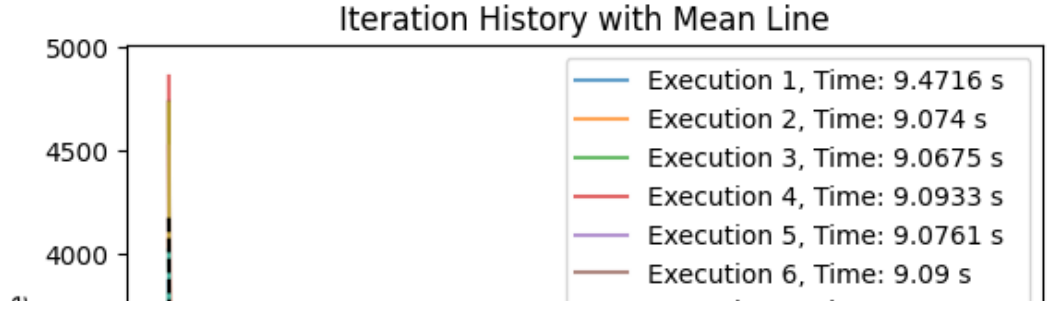


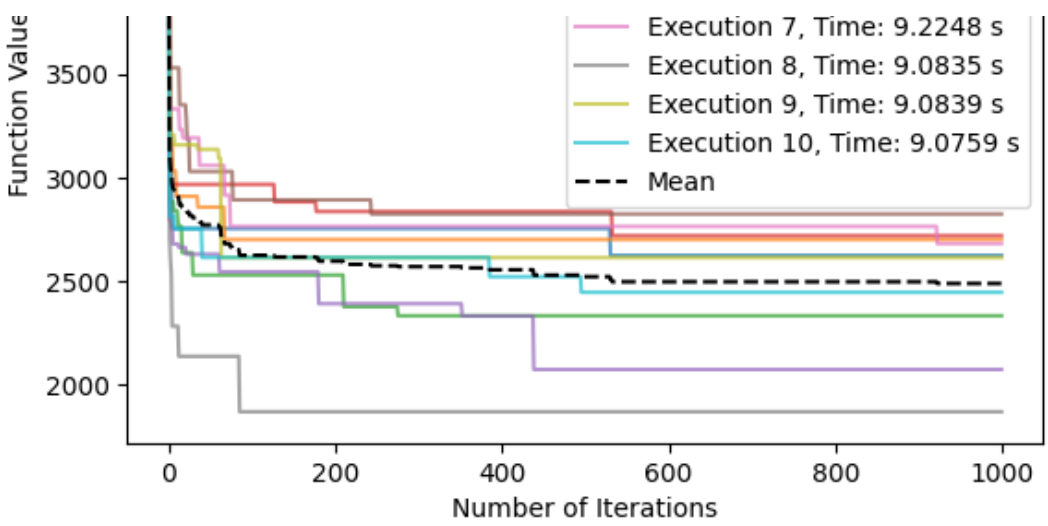



```
14/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```

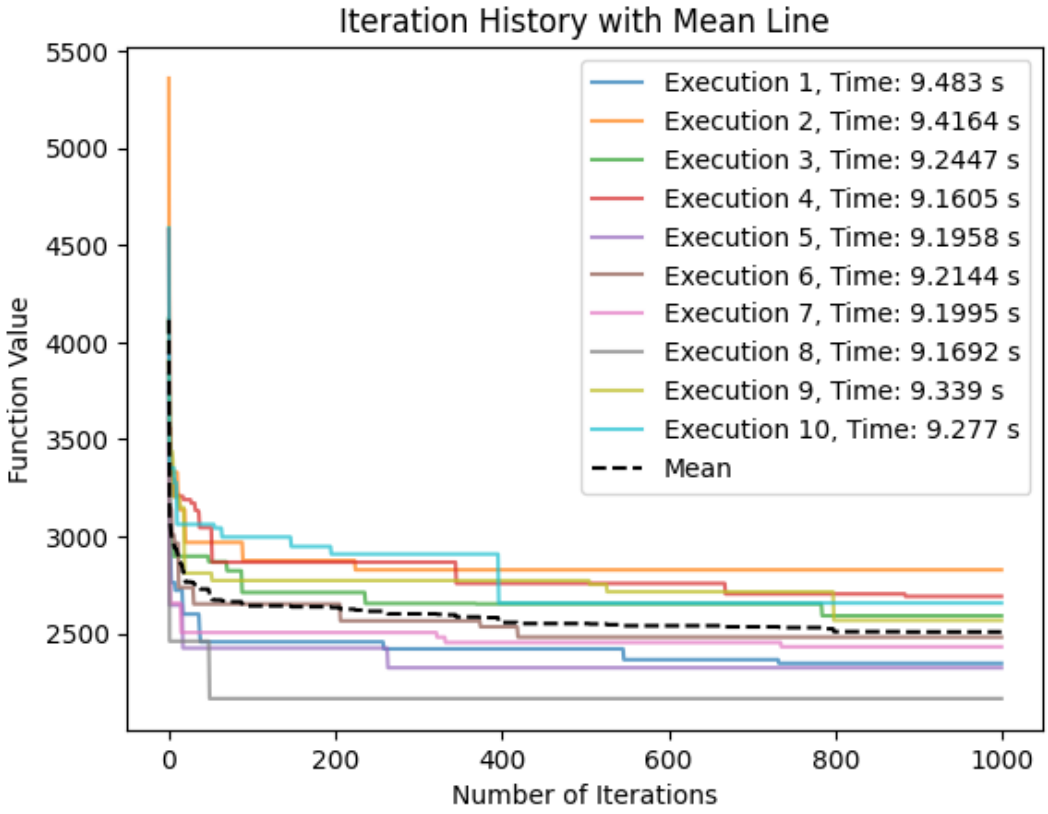



```
15/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

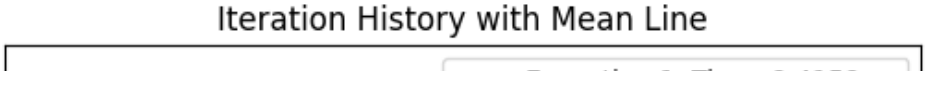


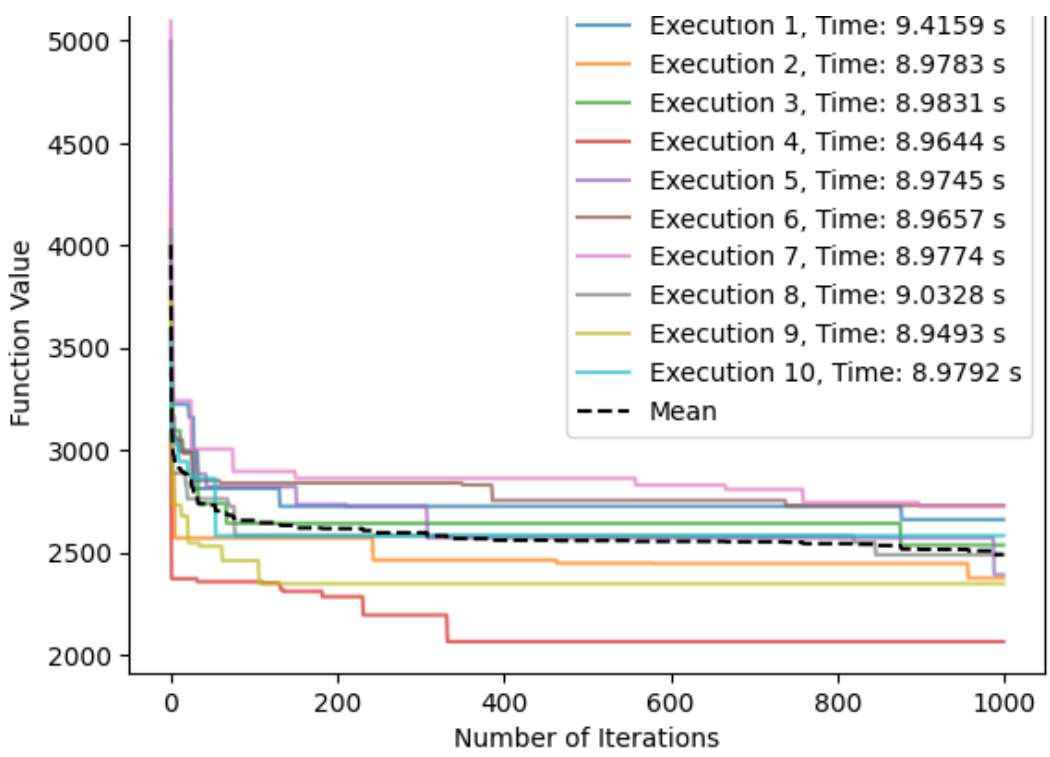


16/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)

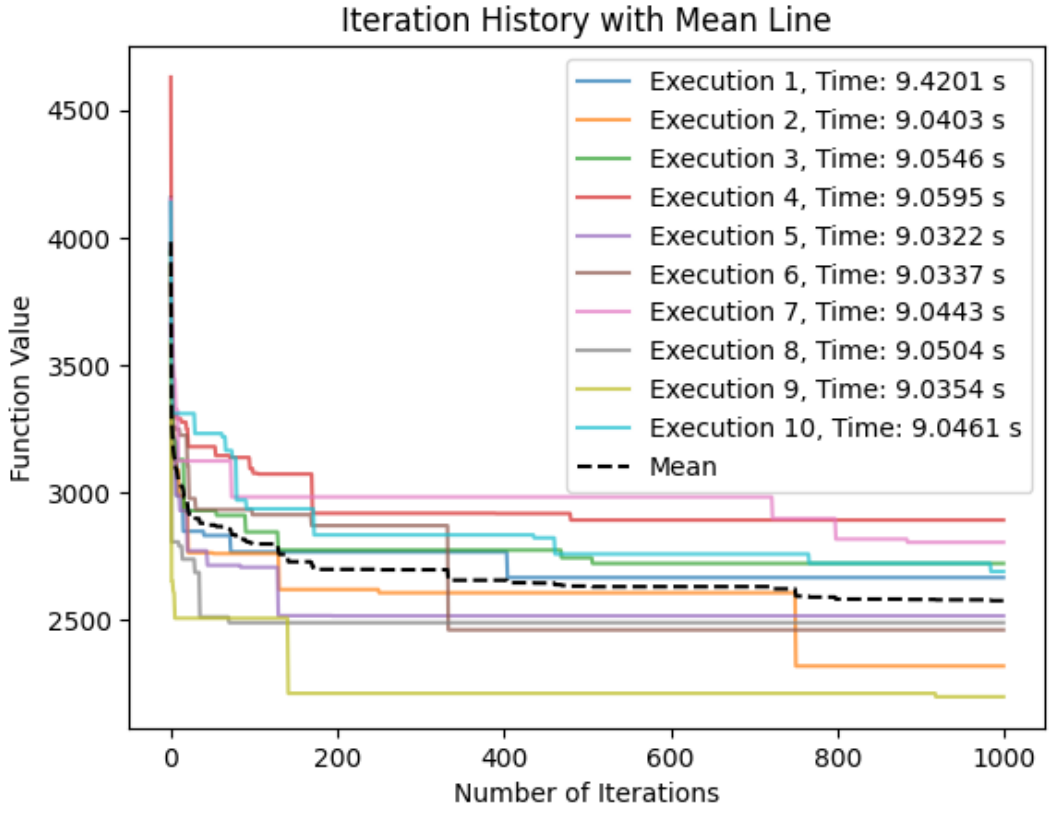


17/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)



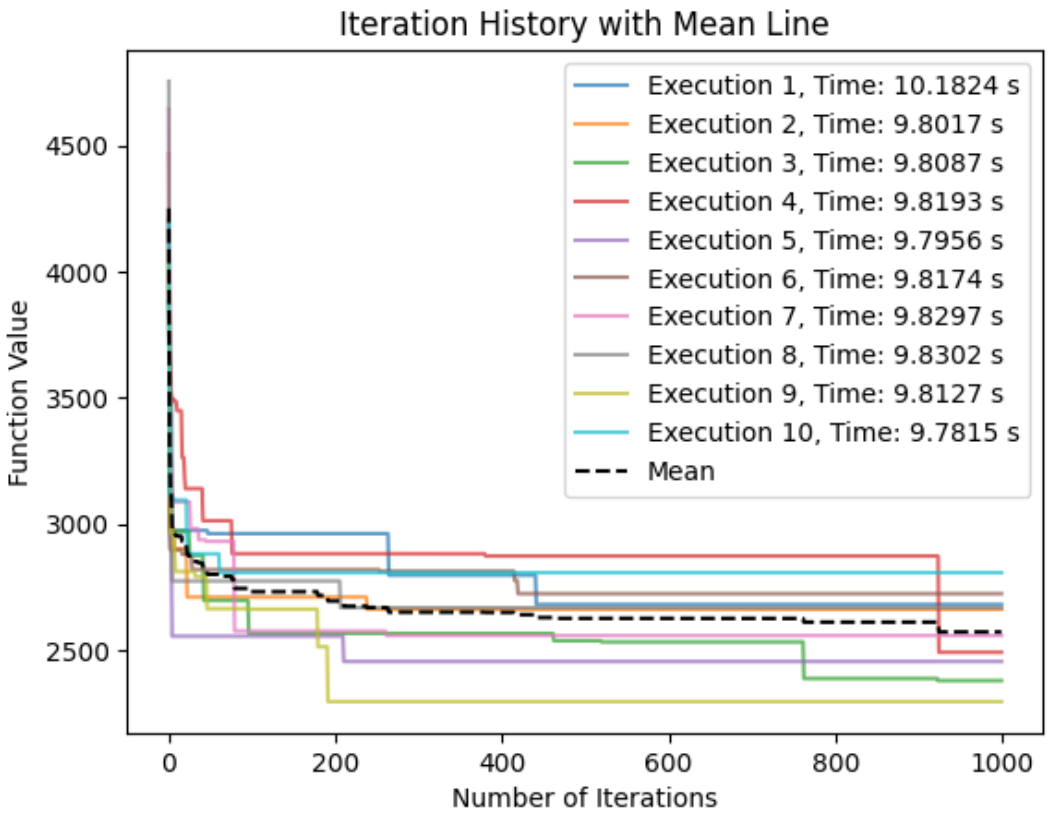


18/24
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)

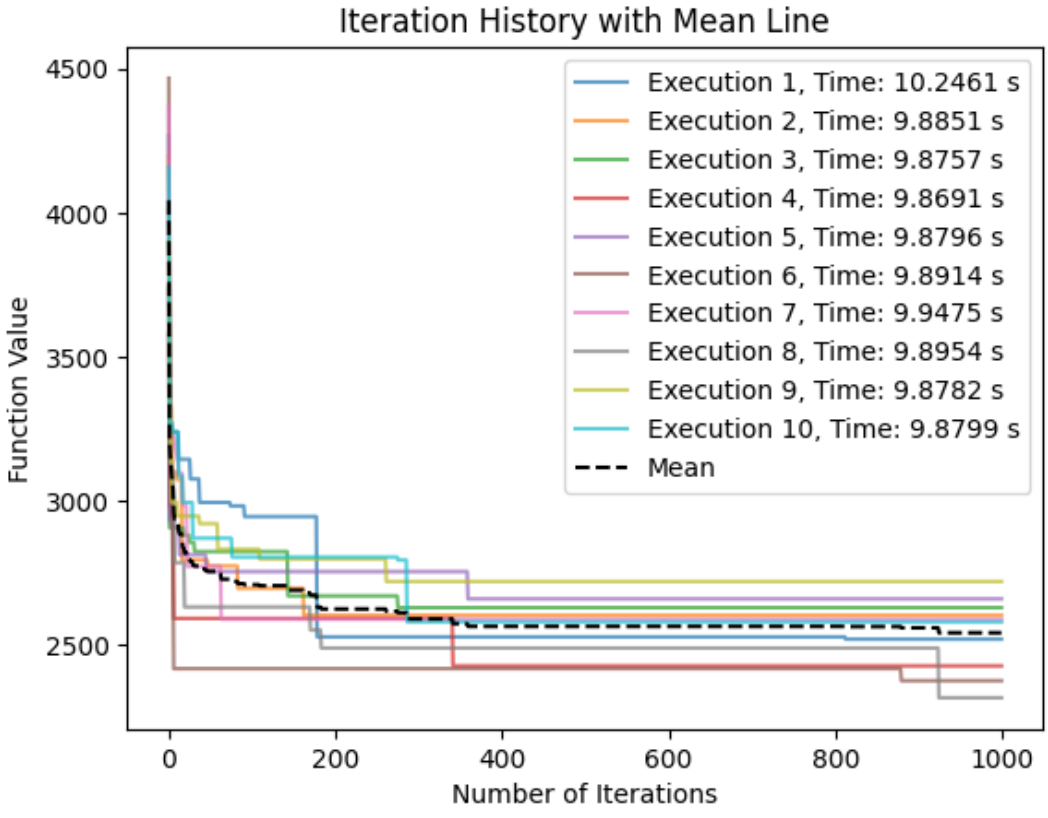


19/24
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
    succession_method: generational_succession

```
    stop_conditions:
        max_iterations(1000)
```

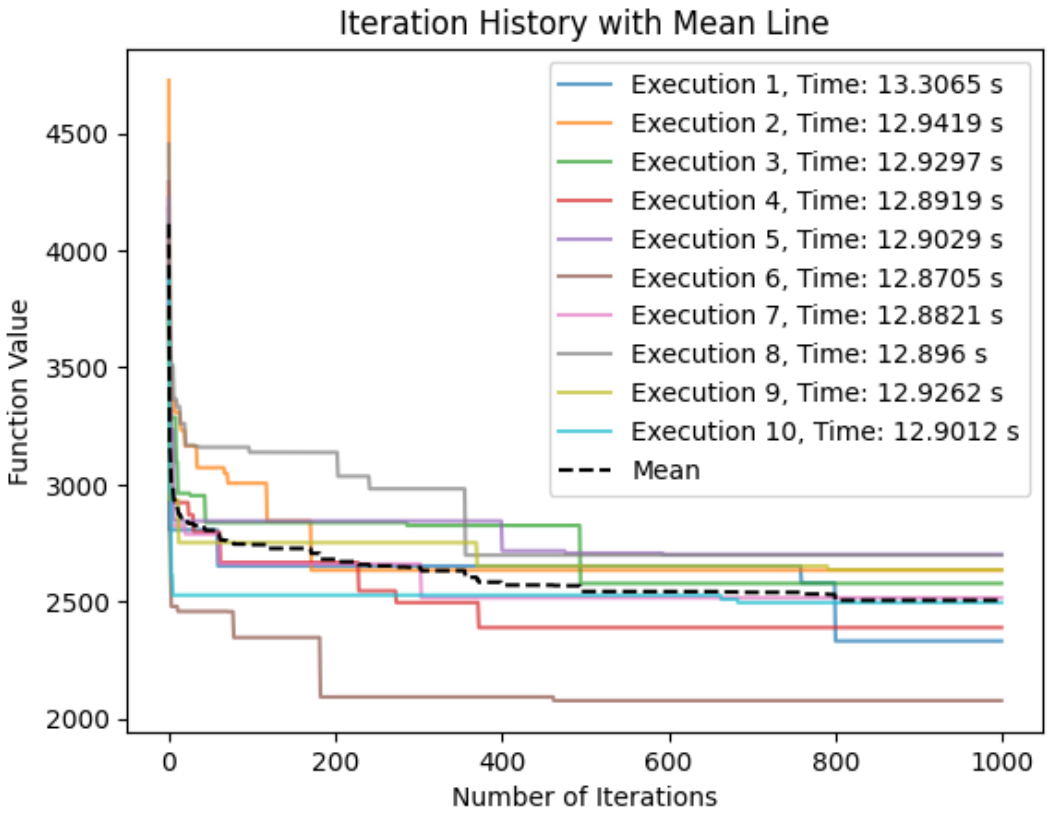



```
20/24
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```



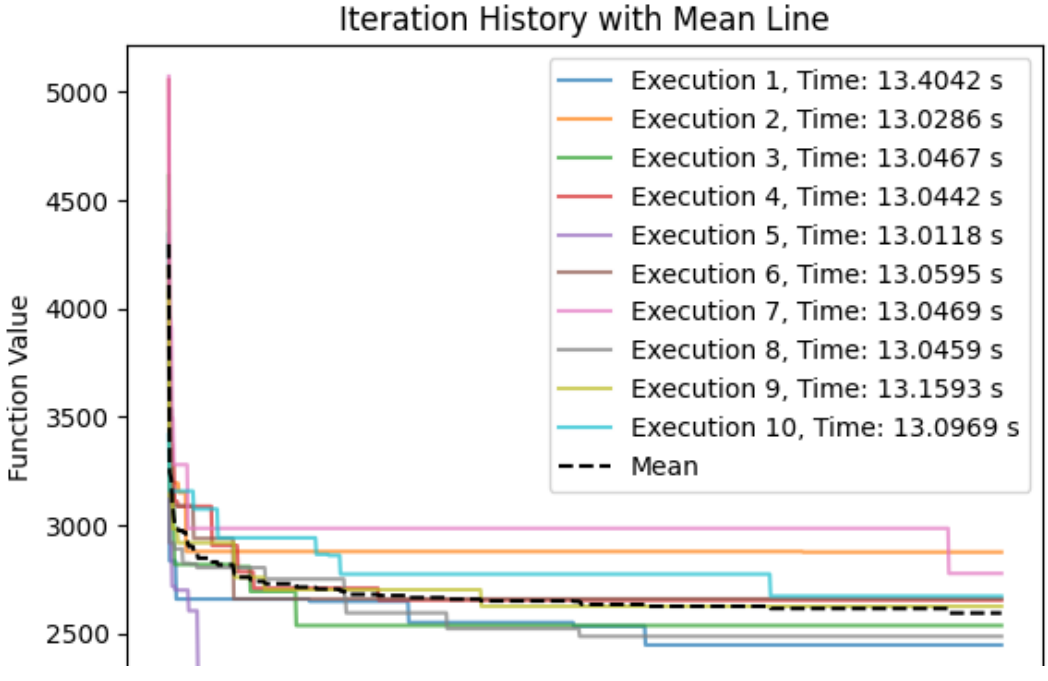

```
21/24
Params:
```

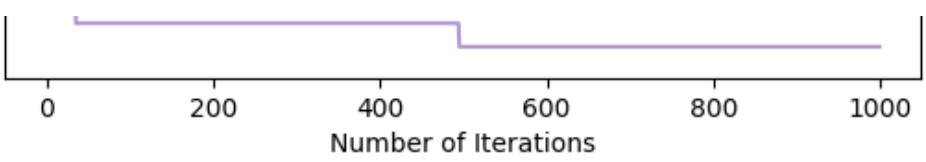
```
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```



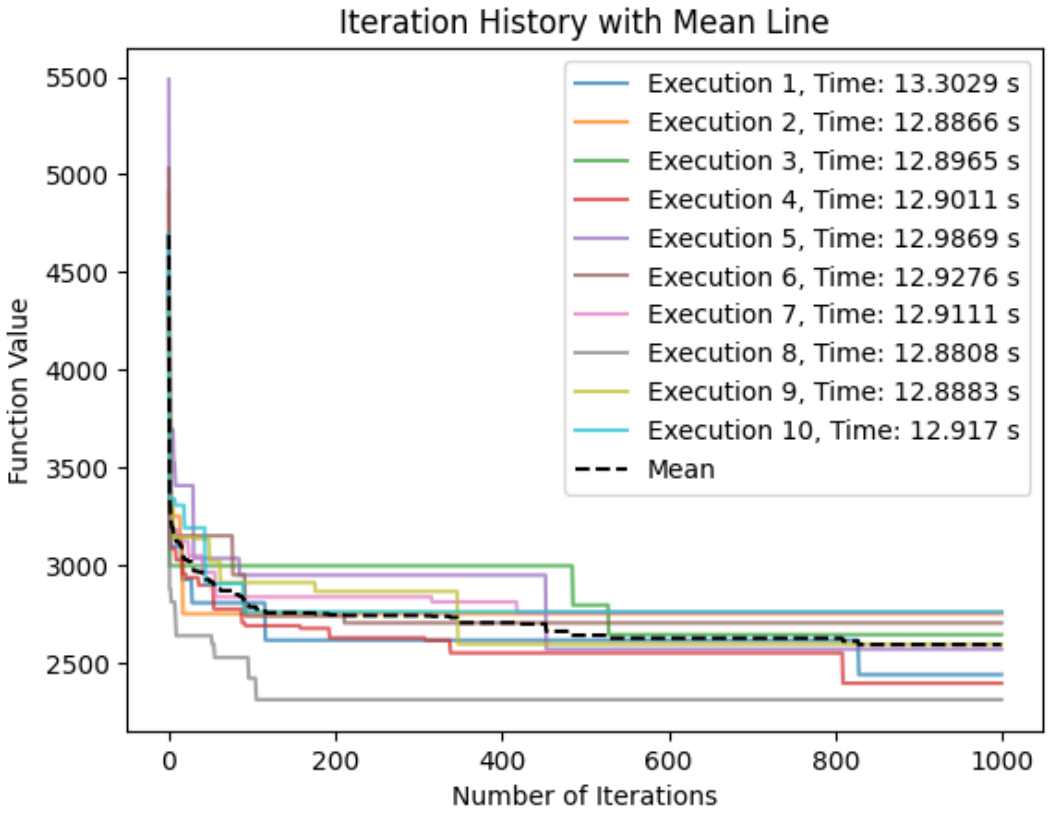
22/24

```
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```

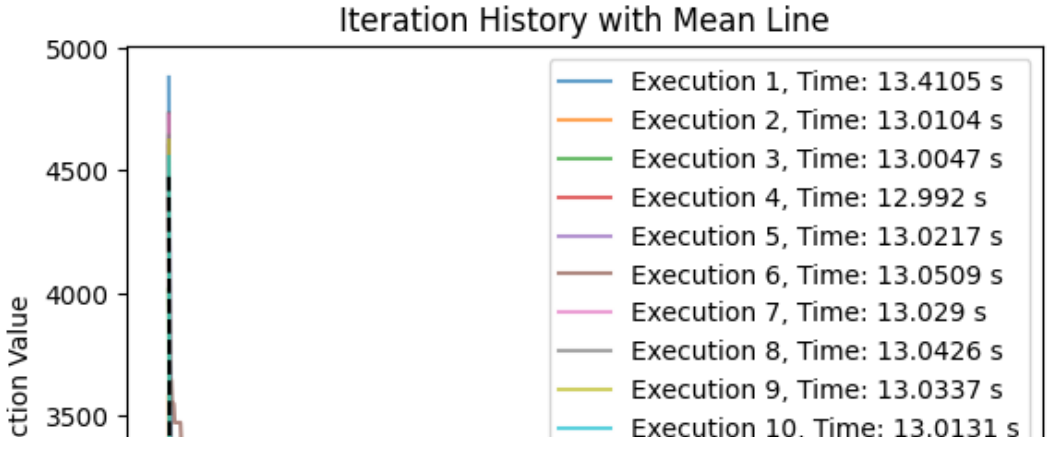


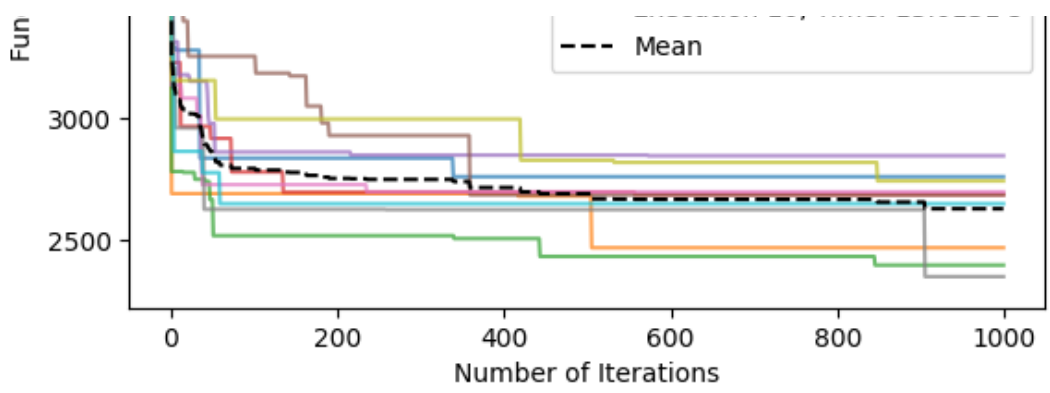



```
23/24
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```

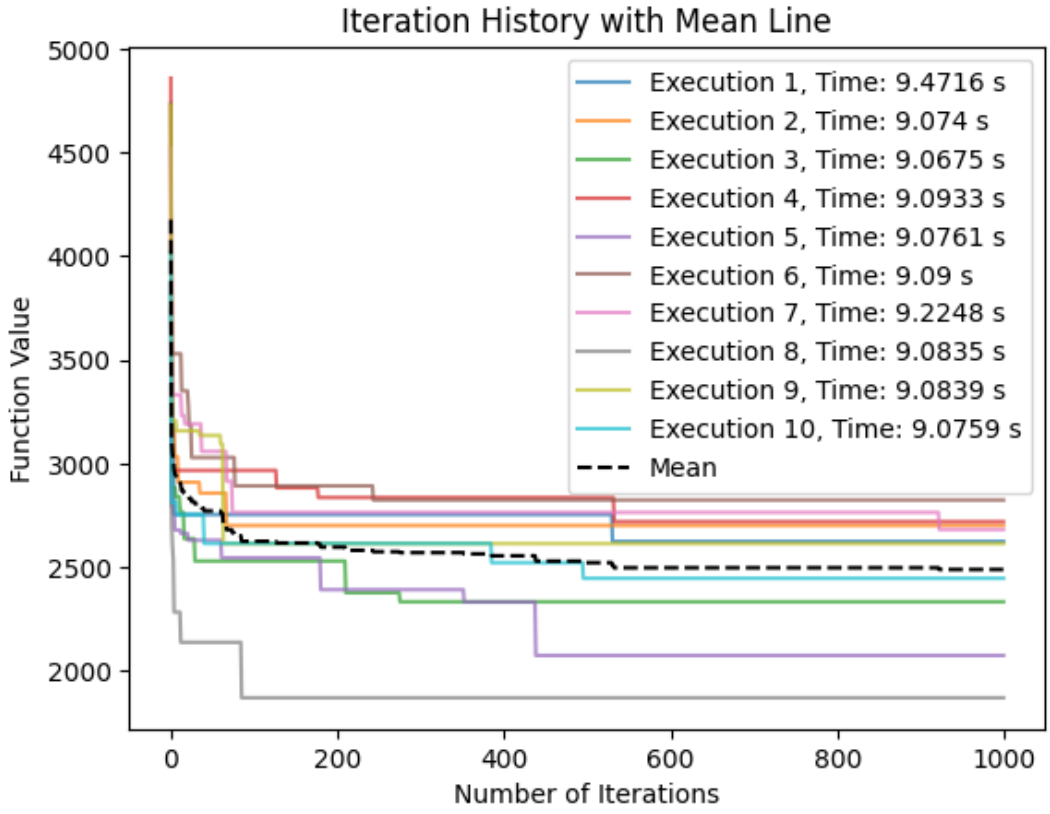



```
24/24
Params:
    population_size: 100
    selection_method: tournament_selection(5)
    genetic_operations:
        mutation
        tsp_crossover(0.7)
    succession_method: elitism_succession(1)
    stop_conditions:
        max_iterations(1000)
```






```
Best config:
Params:
    population_size: 100
    selection_method: tournament_selection(2)
    genetic_operations:
        mutation
        tsp_crossover(0.5)
    succession_method: generational_succession
    stop_conditions:
        max_iterations(1000)
```



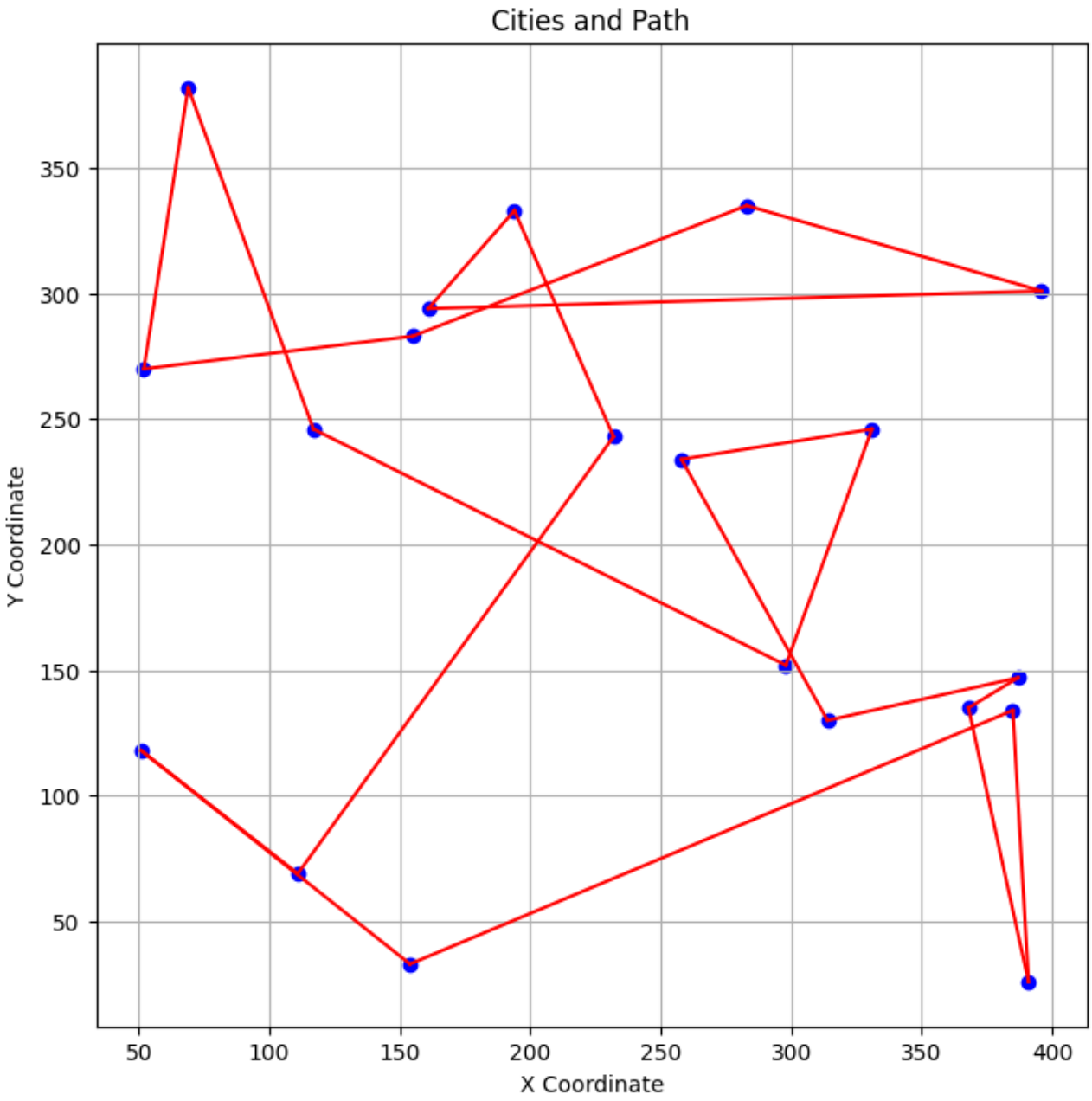

```
Średnia wartość funkcji kosztu: 2488.7261801392283
```

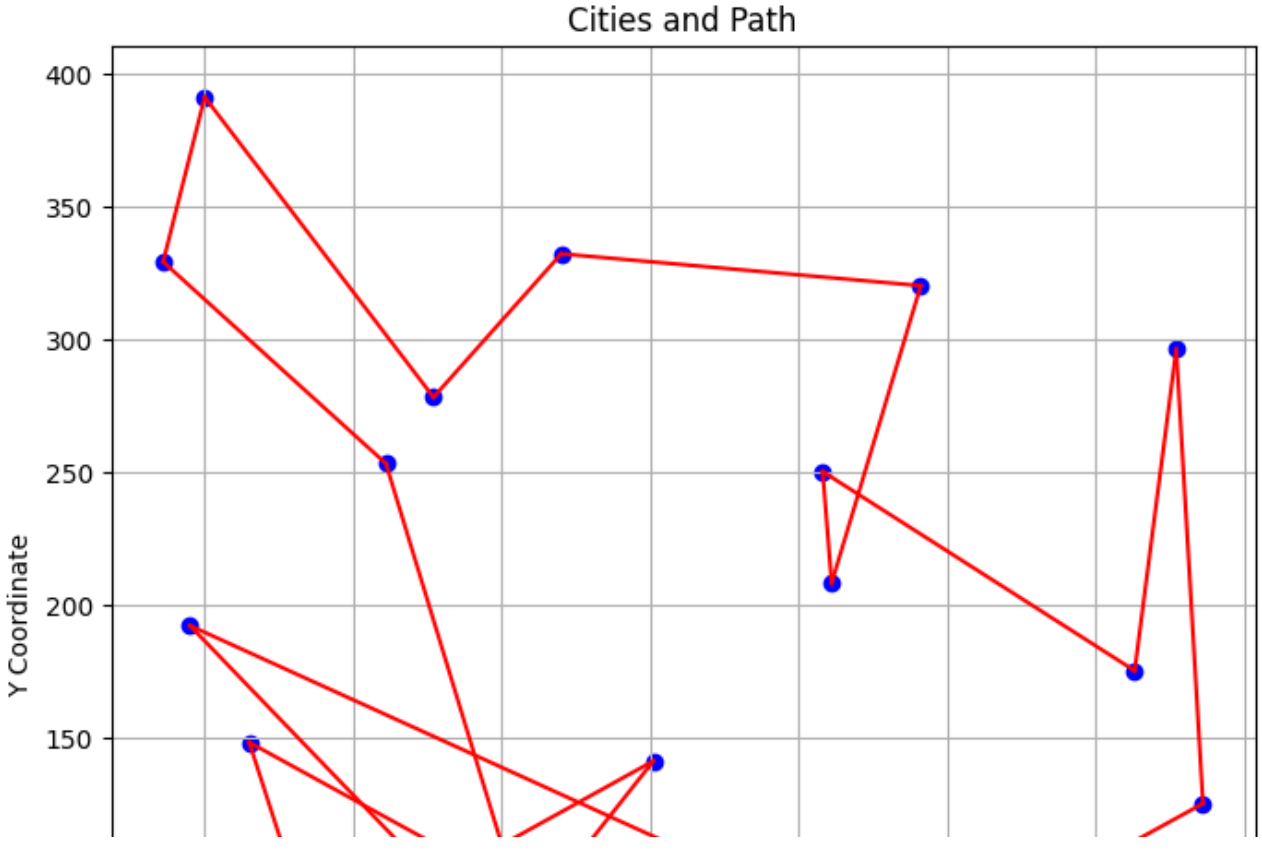
Problem komwojażera na zoptimizowanym algorytmie ewolucyjnym

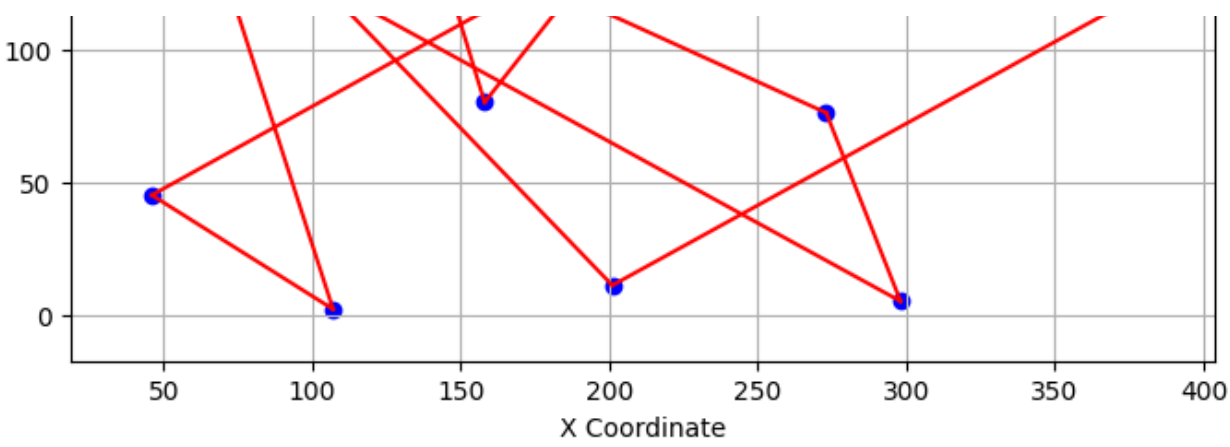
In [9]:

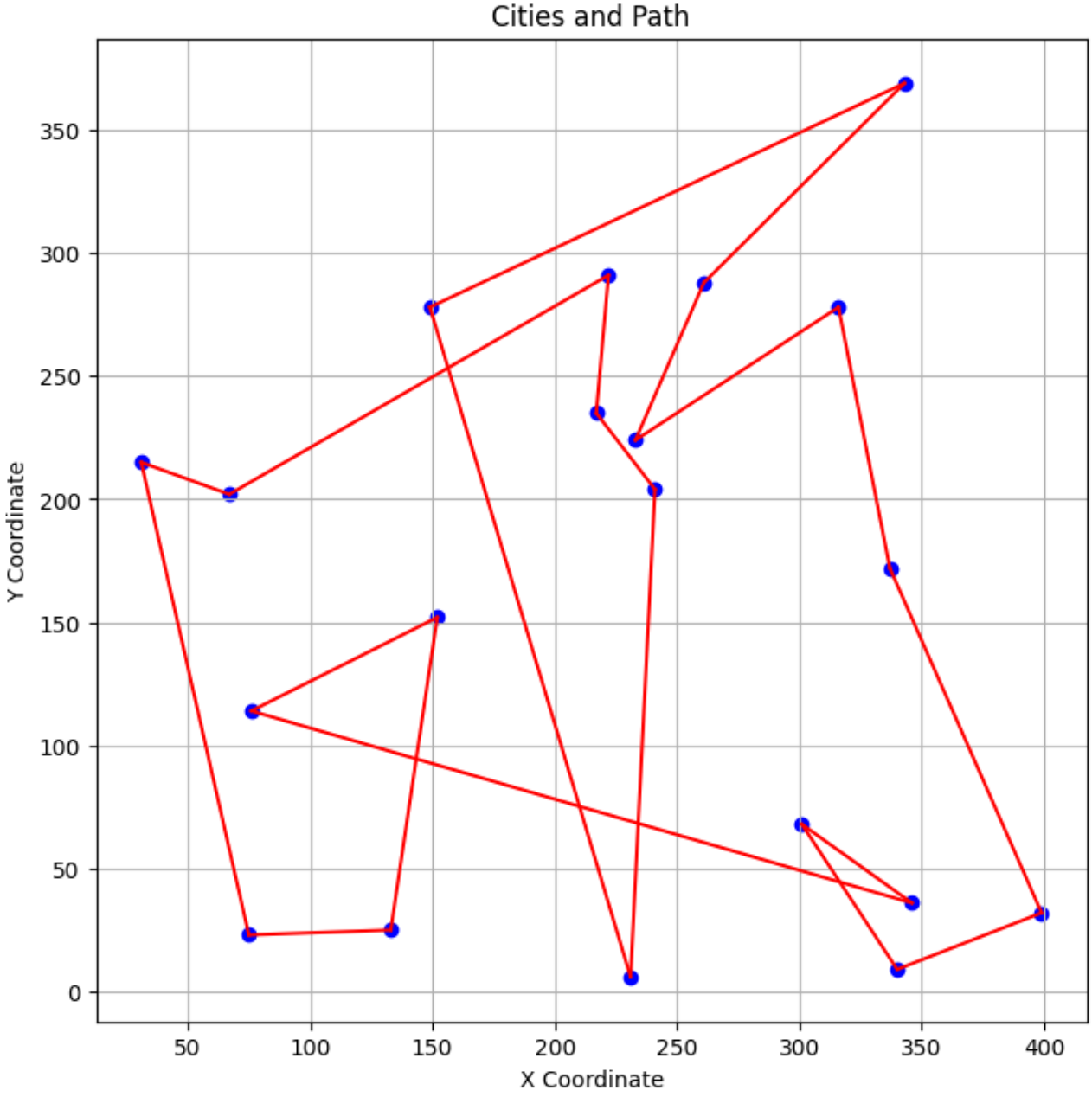
```
solver_opt = EvoSolver(
    individual_type=TSPIndividualType(n_genes=len(cities)),
    population_size=best_params['population_size'],
    selection_method=best_params['selection_method'],
    genetic_operations=best_params['genetic_operations'],
    succession_method=best_params['succession_method'],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

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

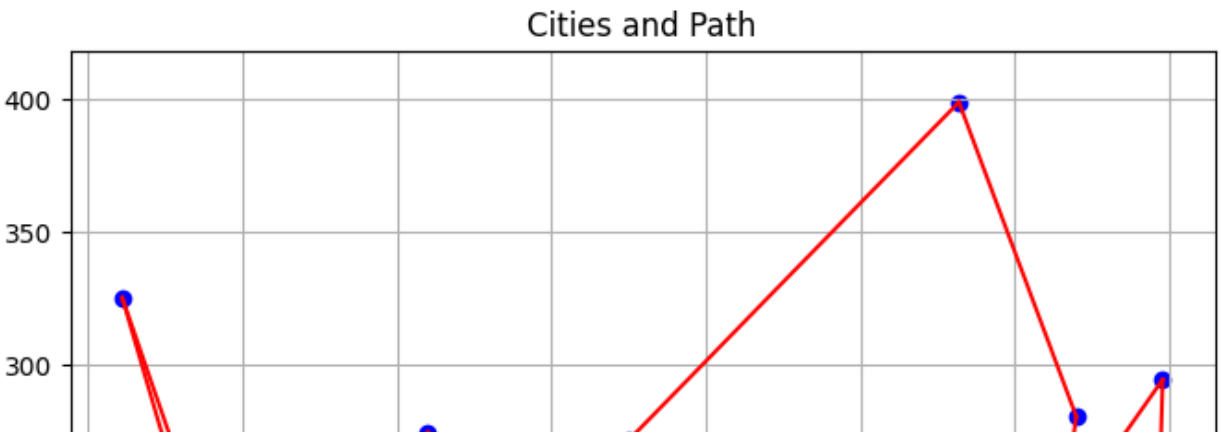
```
plot_cities(cities2, result.x)
```

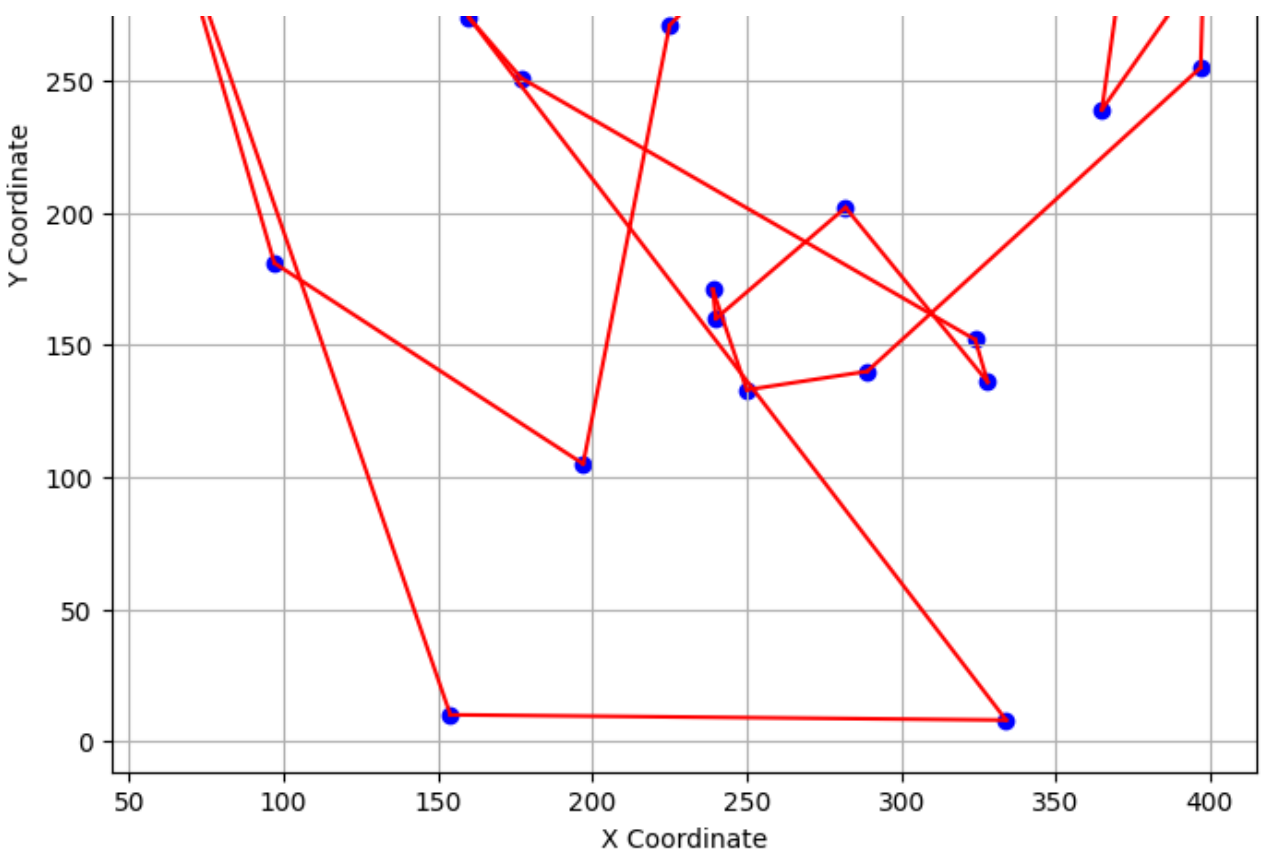





100
50
0
50
100
150
200
250
300
350
400
X Coordinate


Cities and Path
350
300
250
200
150
100
50
0
Y Coordinate
50
100
150
200
250
300
350
400
X Coordinate


Cities and Path
400
350
300



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 metoda sukcesji okazała się być sukcesja generacyjna - prawdopodobnie zwiększa to zdolność eksploracyjną algorytmu**