Genetic Algorithm

```
In [1]: import francium.algorithms.genetic_algorithm as ga
        import francium.core.eval_functions as eval_functions
```

using an environment with z = x^2 + y^2

```
In [4]: agent = ga.Agent()
        env = ga.Environment(x_bounds=(-5.0, 5.0), y_bounds=(-5.0, 5.0), eval_func=eval_functions.convex_x_square)
        solver = ga.Solver(agent=agent, environment=env, pop_size=100)
```

```
In [5]: solver.init_solver()
```
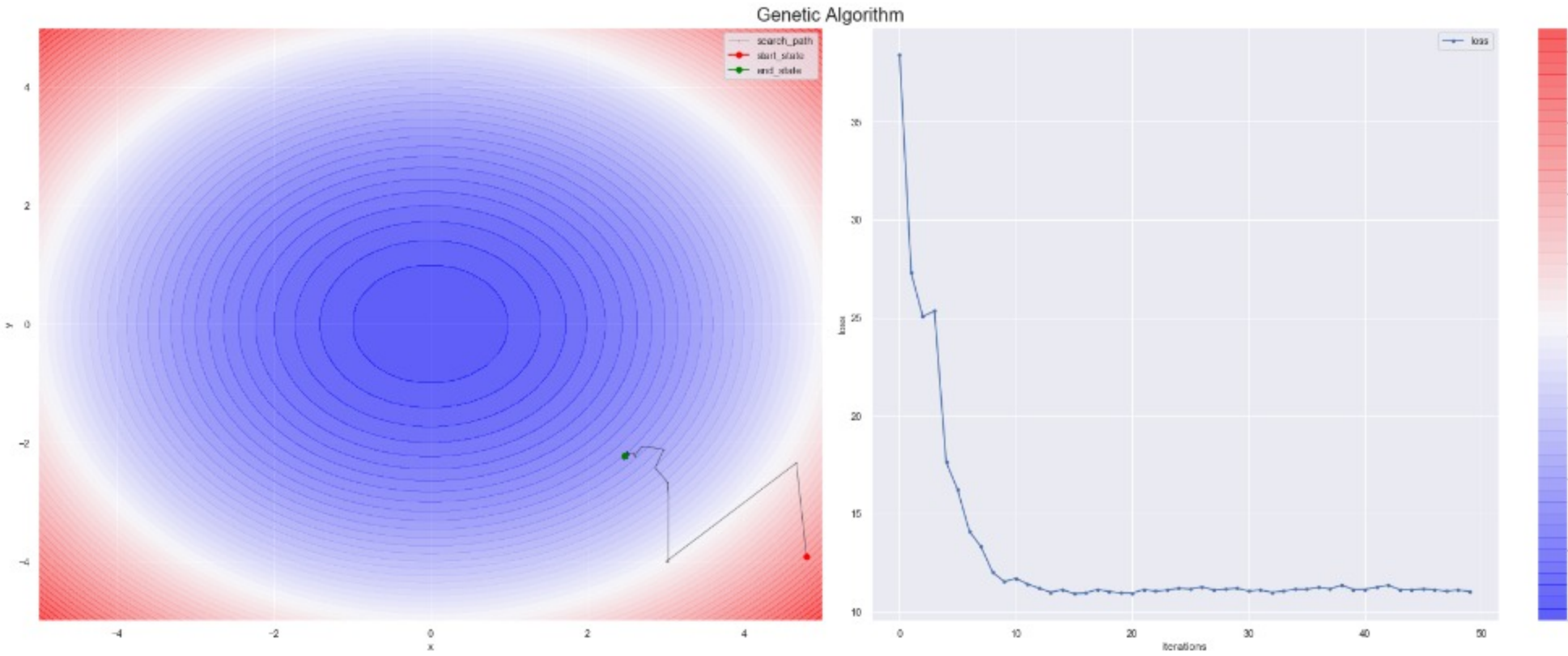
```
[ 2020-12-03 16:03:30,189 - francium.algorithms.genetic_algorithm.solver ] INFO: => Initialized Agent !
```

```
In [6]: for episode in range(50):
            trainable = solver.train_step()
            if not trainable:
                break
```
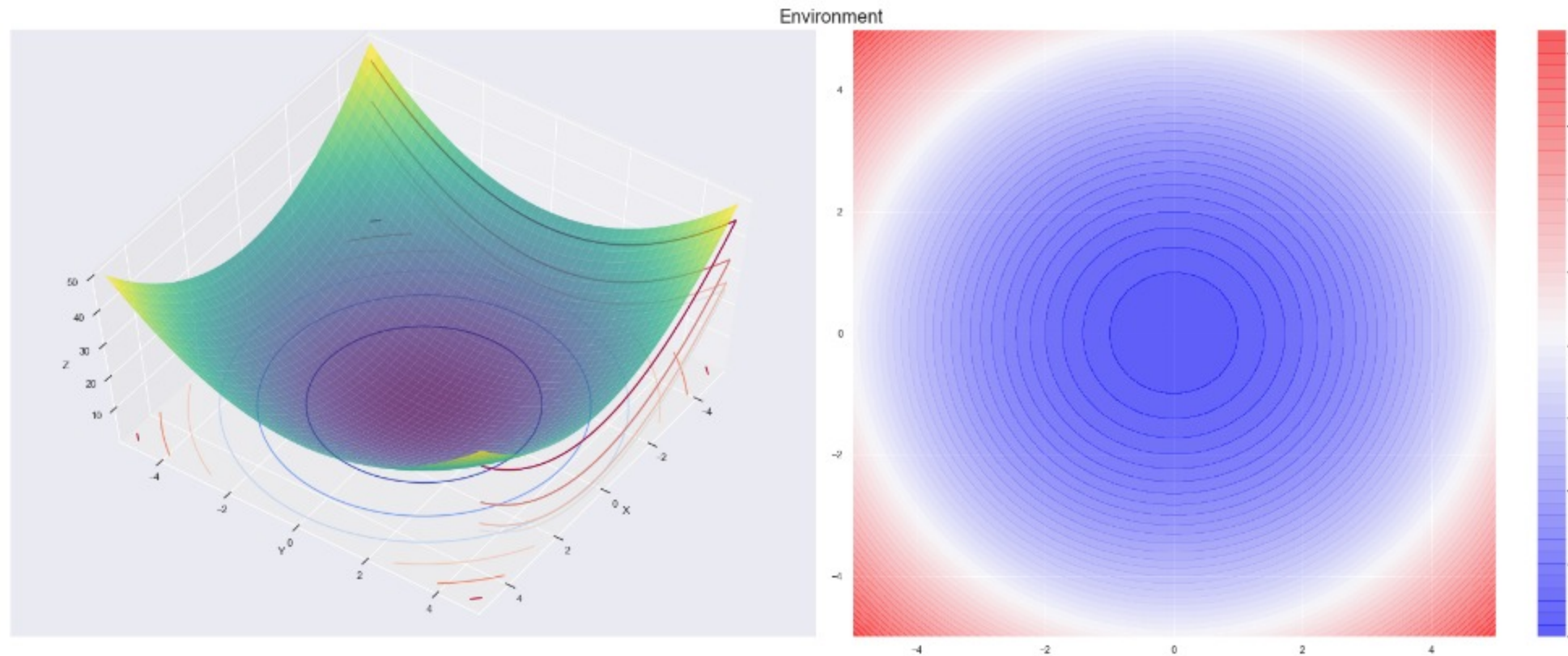
```
In [7]: solver.memory.best_episode
```

```
Out[7]: {'x': 2.5146588957311944, 'y': -2.1445624334917754, 'z': 10.922657393024195}
```

```
In [8]: solver.plot_history()
```



```
In [9]: env.plot_environment()
```


using an environment with z = 5 * sin(x^2 + y^2) + x^2 + y^2

```
In [11]: agent = ga.Agent()
         env = ga.Environment(x_bounds=(-5.0, 5.0), y_bounds=(-5.0, 5.0), eval_func=eval_functions.sinx_plus_x)
         solver = ga.Solver(agent=agent, environment=env, pop_size=100)
```

```
In [12]: solver.init_solver()
```
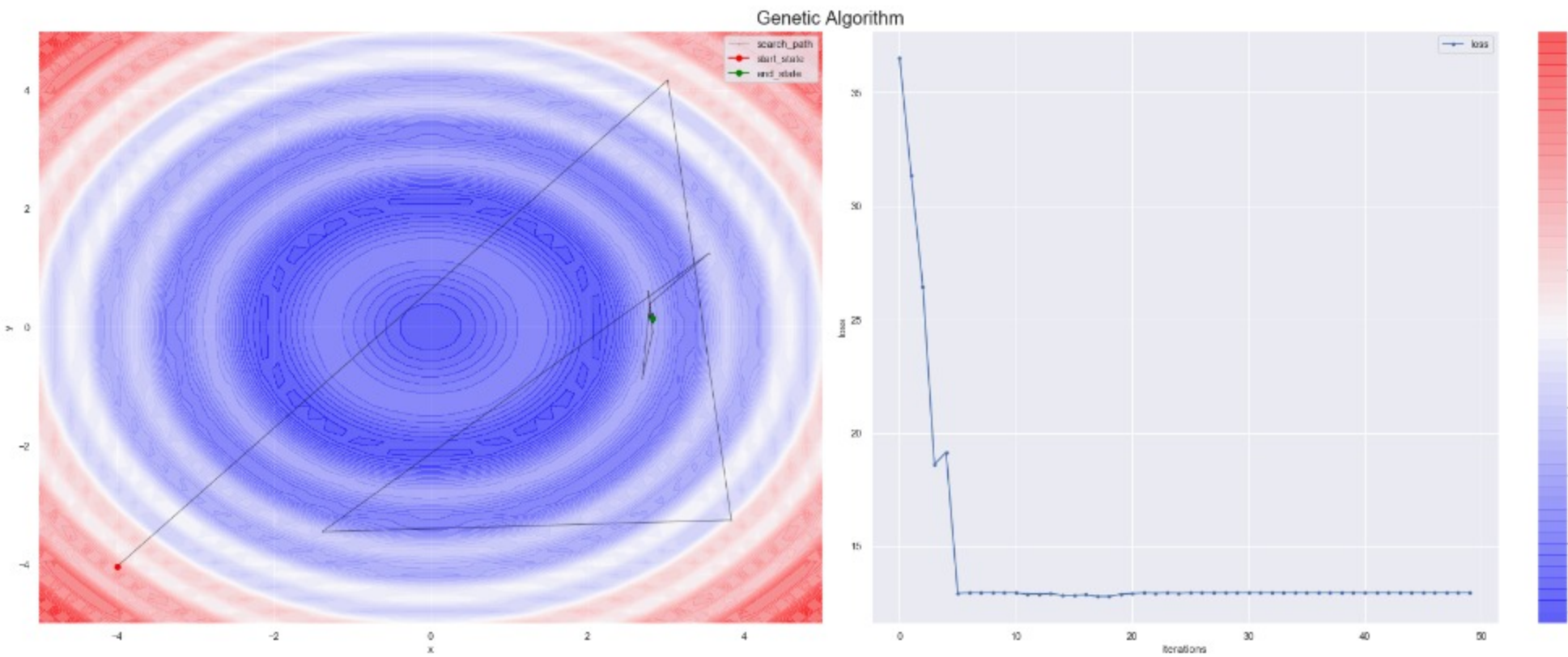
```
[ 2020-12-03 16:06:02,604 - francium.algorithms.genetic_algorithm.solver ] INFO: => Initialized Agent !
```

```
In [13]: for episode in range(50):
             trainable = solver.train_step()
             if not trainable:
                 break
```

```
In [14]: solver.memory.best_episode
```

```
Out[14]: {'x': 2.7889714146335893, 'y': 0.15665519229271258, 'z': 12.796381098863423}
```

```
In [15]: solver.plot_history()
```



```
In [16]: env.plot_environment()
```

```
C:\Users\shadowleaf\anaconda3\envs\thetensorclan-aws\lib\site-packages\numpy\core\_asarray.py:136: VisibleDeprecationWarning: C
reating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths
or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
    return array(a, dtype, copy=False, order=order, subok=True)
```