Assignment #2
Genetic Programming

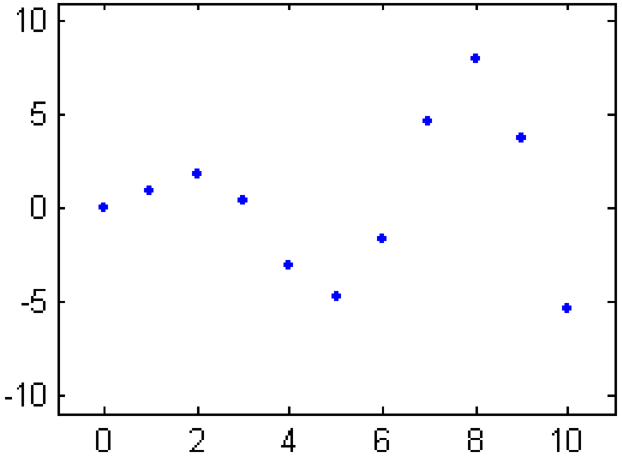
MECS 4510

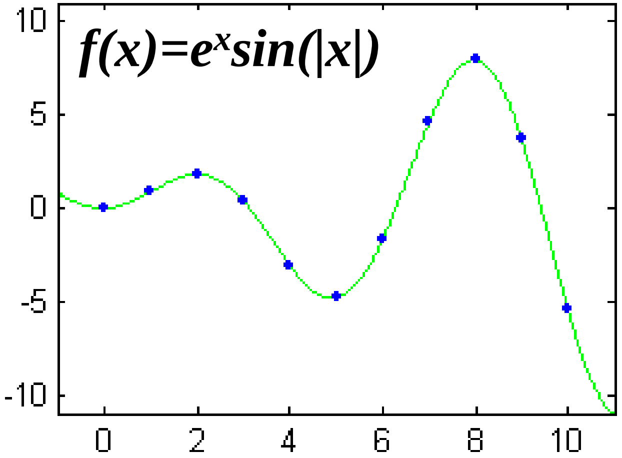
Evolutionary Computation

Hod Lipson

Symbolic Regression

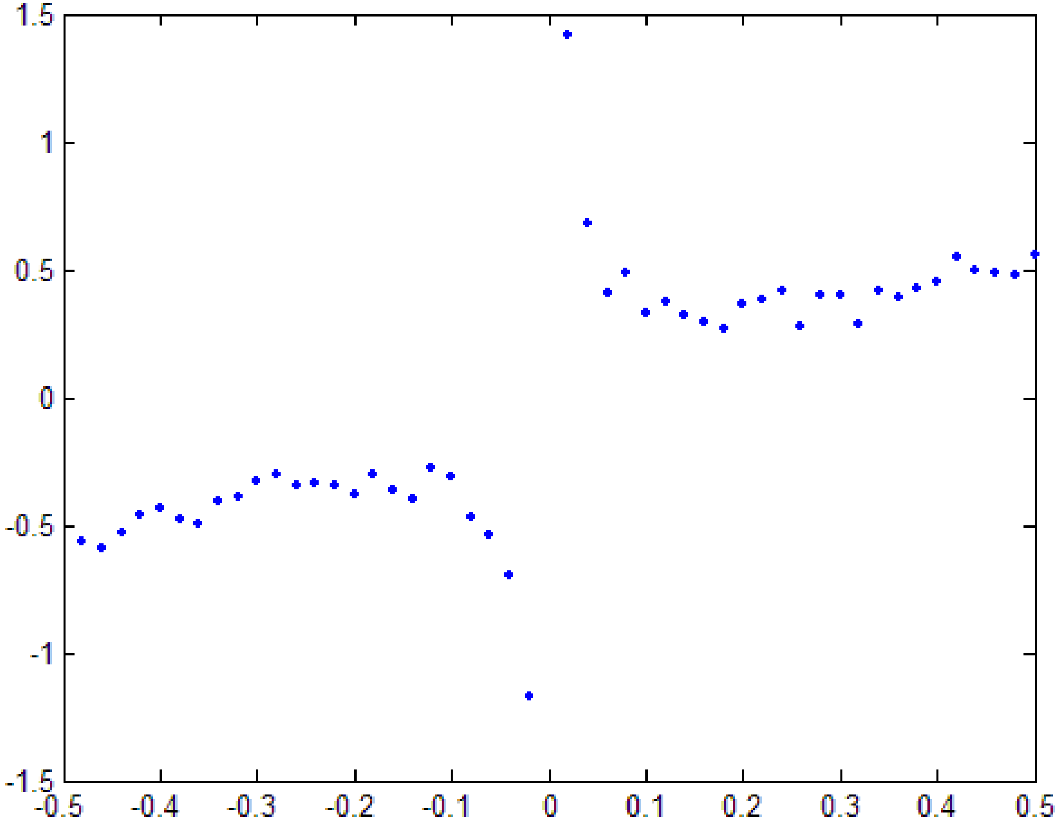
What function describes this data?



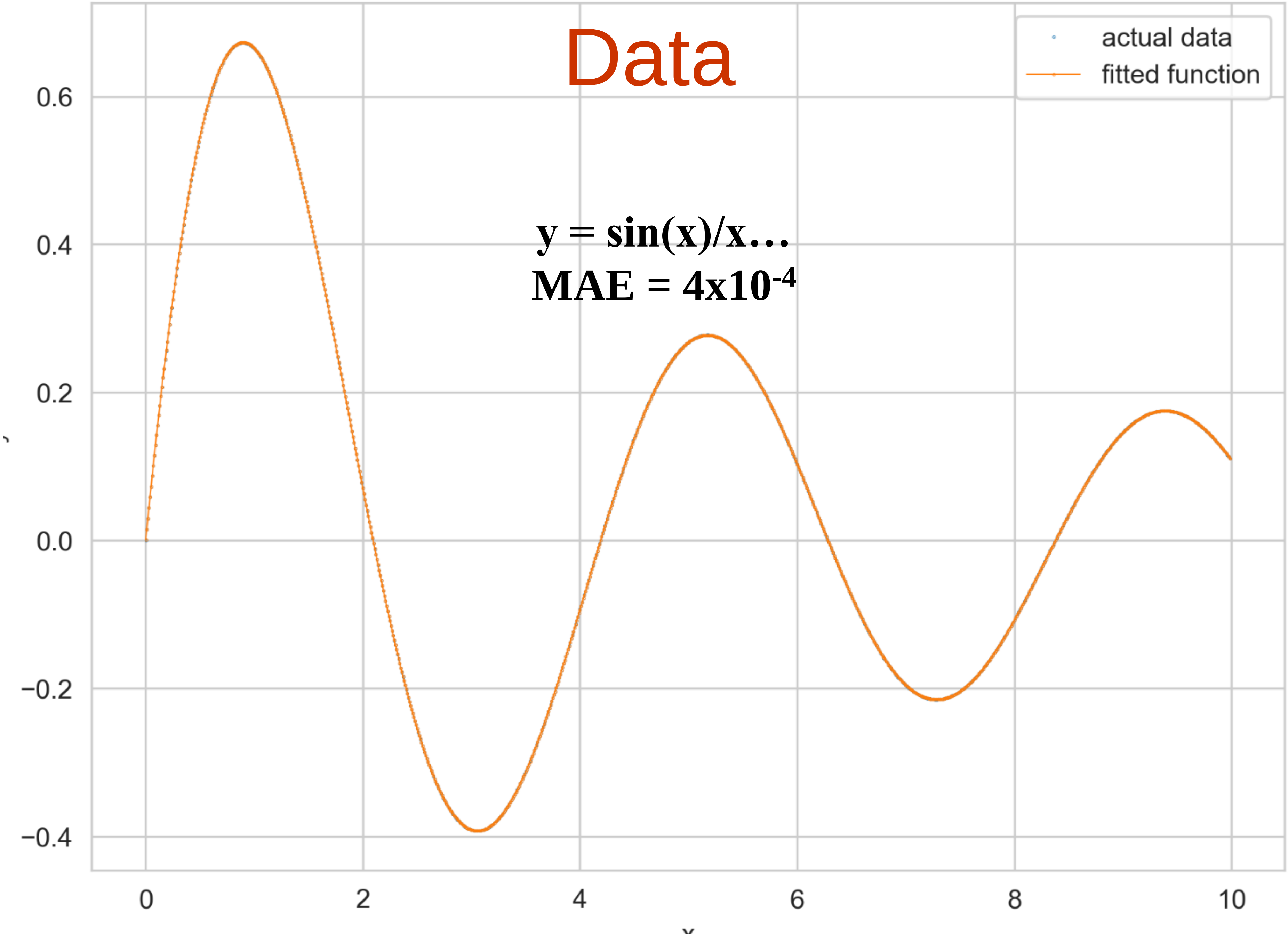


John Koza, 1992

Data



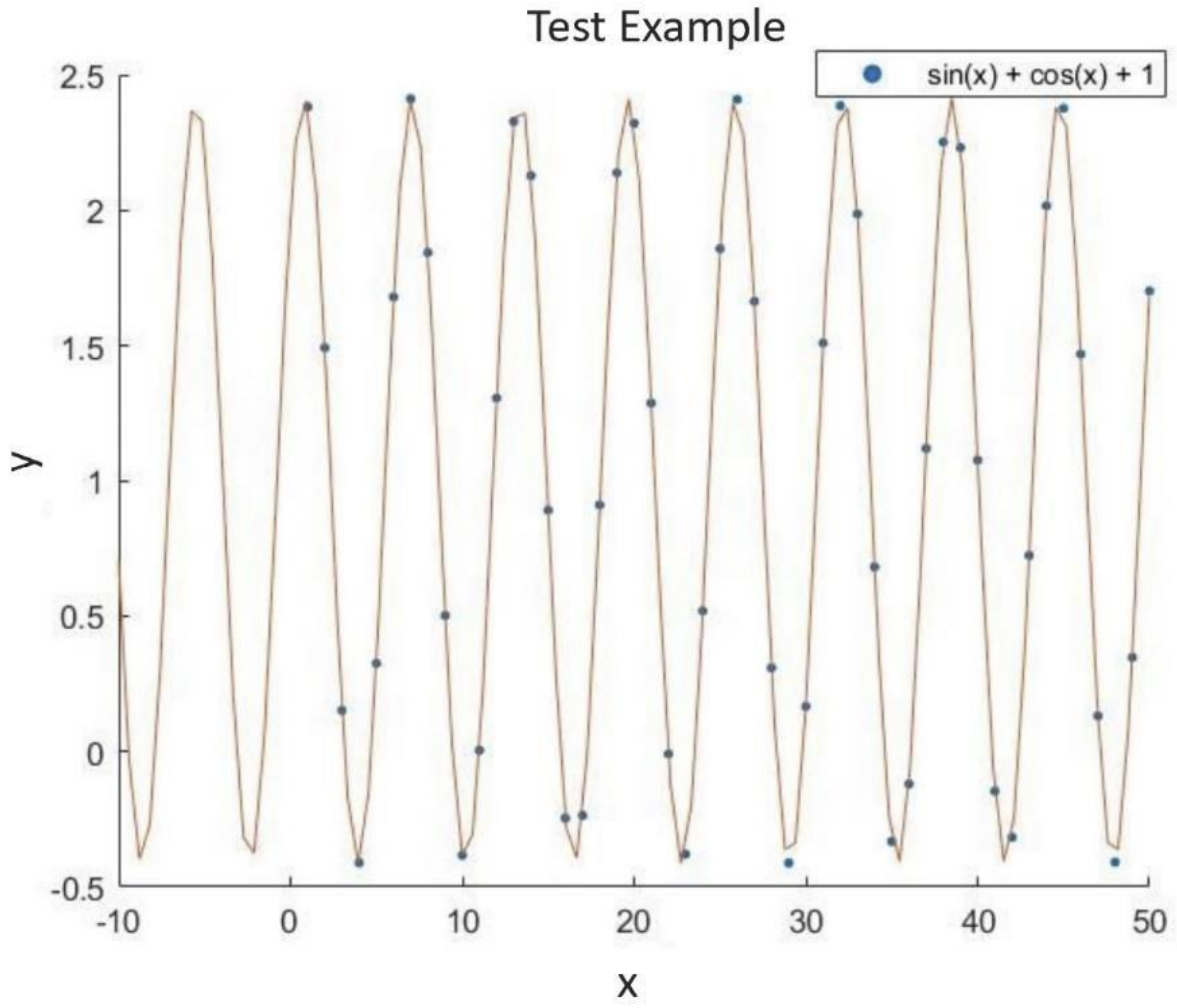
Data



y = sin(x)/x…

MAE = 4x10^{-4}

Boxi Xia

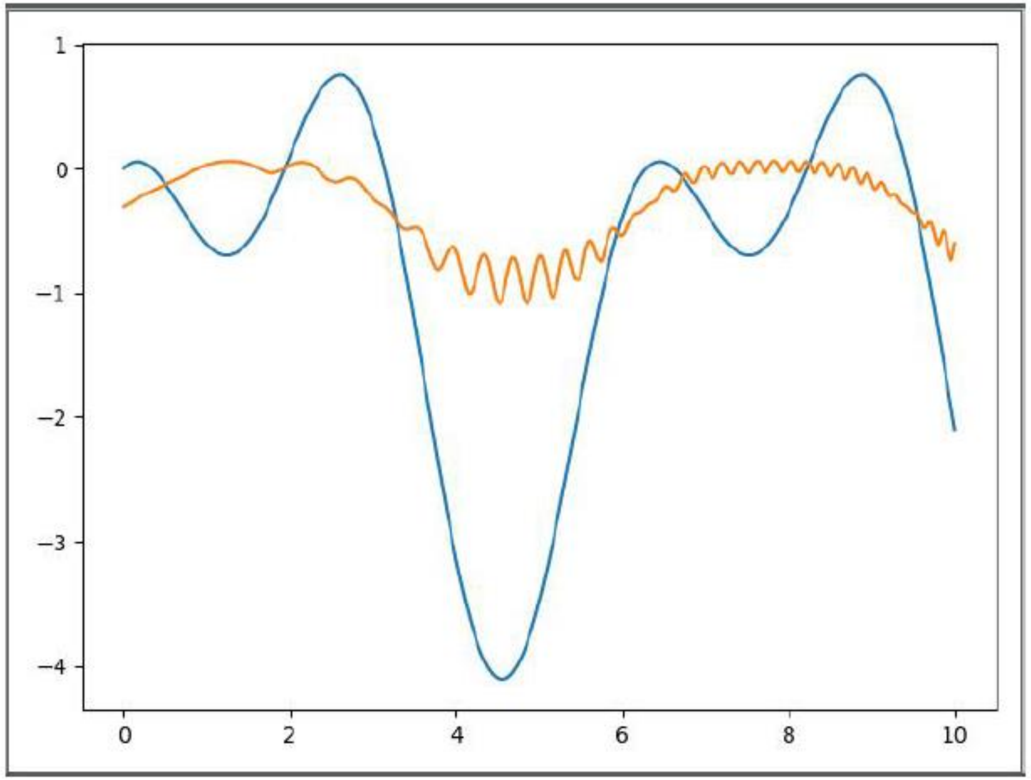


Yaoyao Xu, Ruoyao Zhao

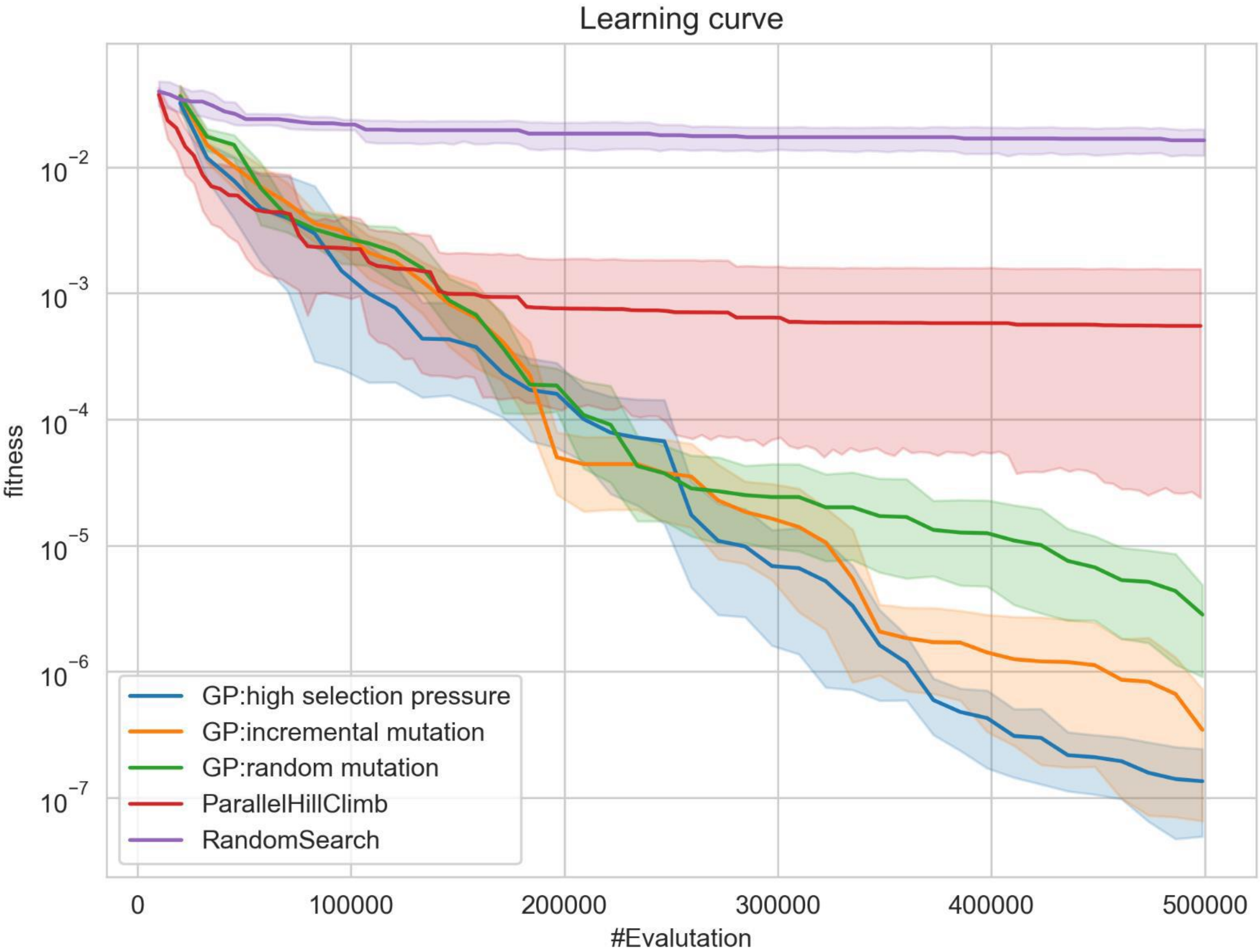
Week 1: Random function

equation = cos(cos(cos(x*x))*sin(sin(x))+
x/x*(-9.247776198625752)+
(-8.62238505749605)/(-4.15149813701656))-
sin((1.2185149095292367))

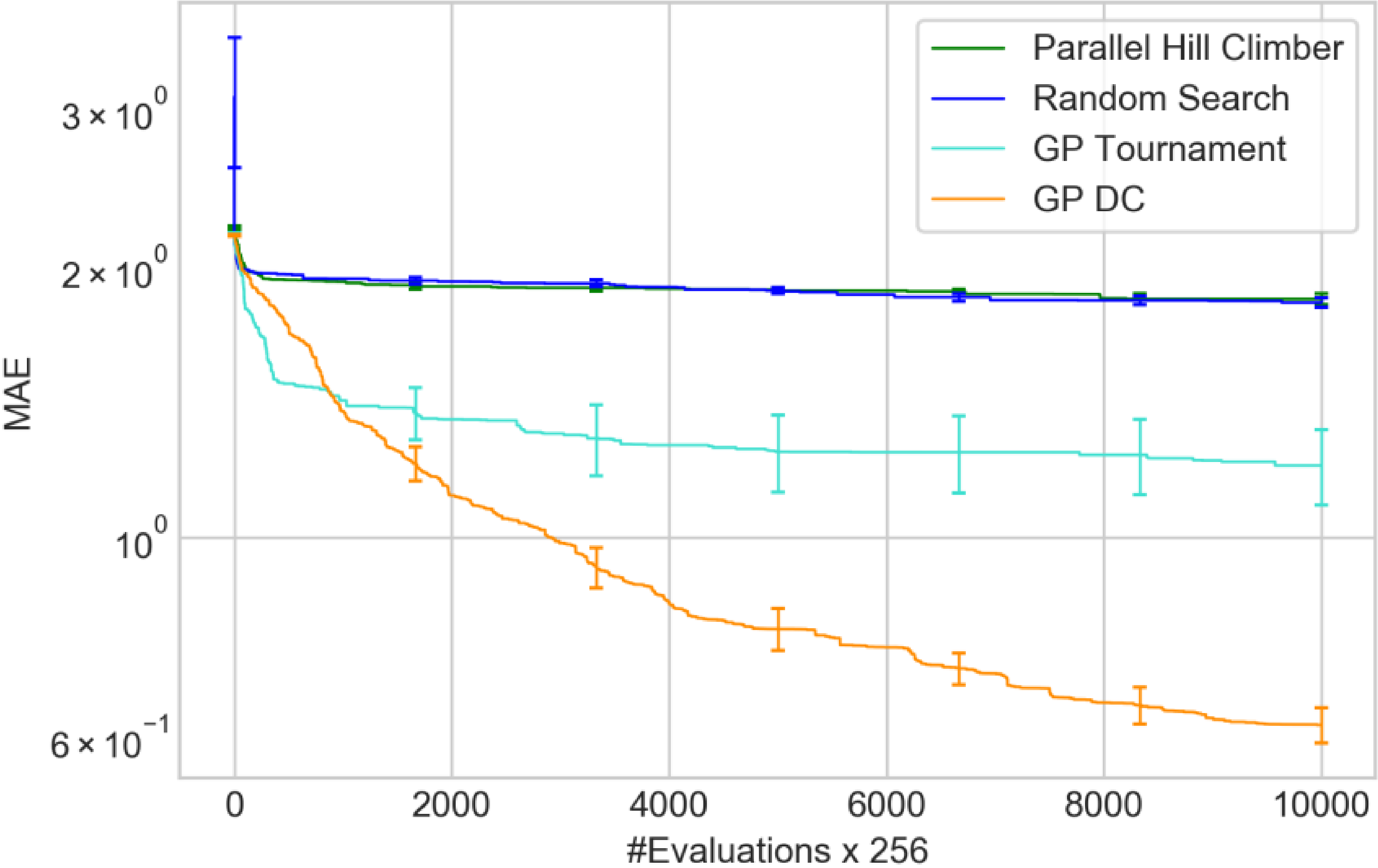
MSE = 1.5227438847355902

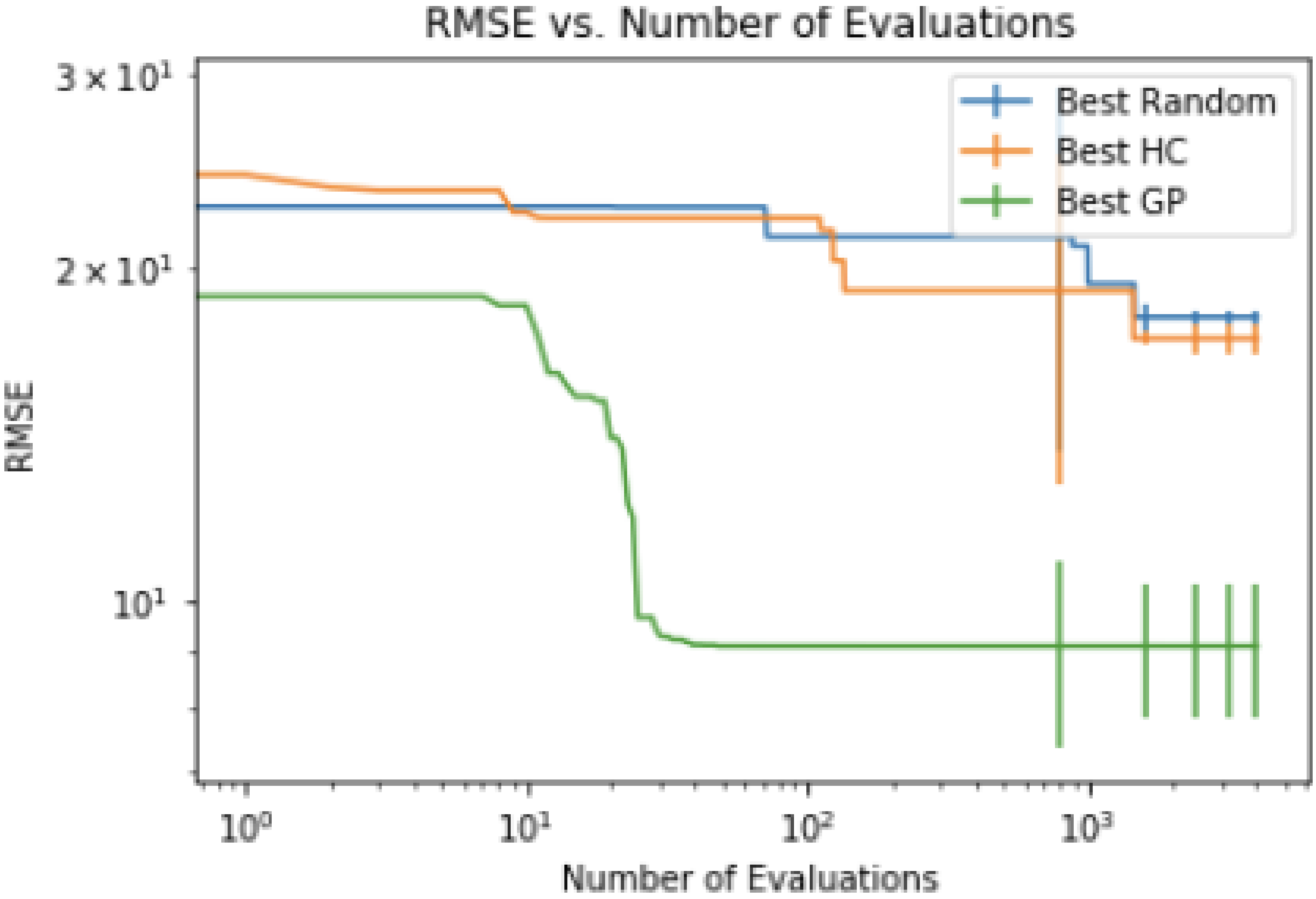


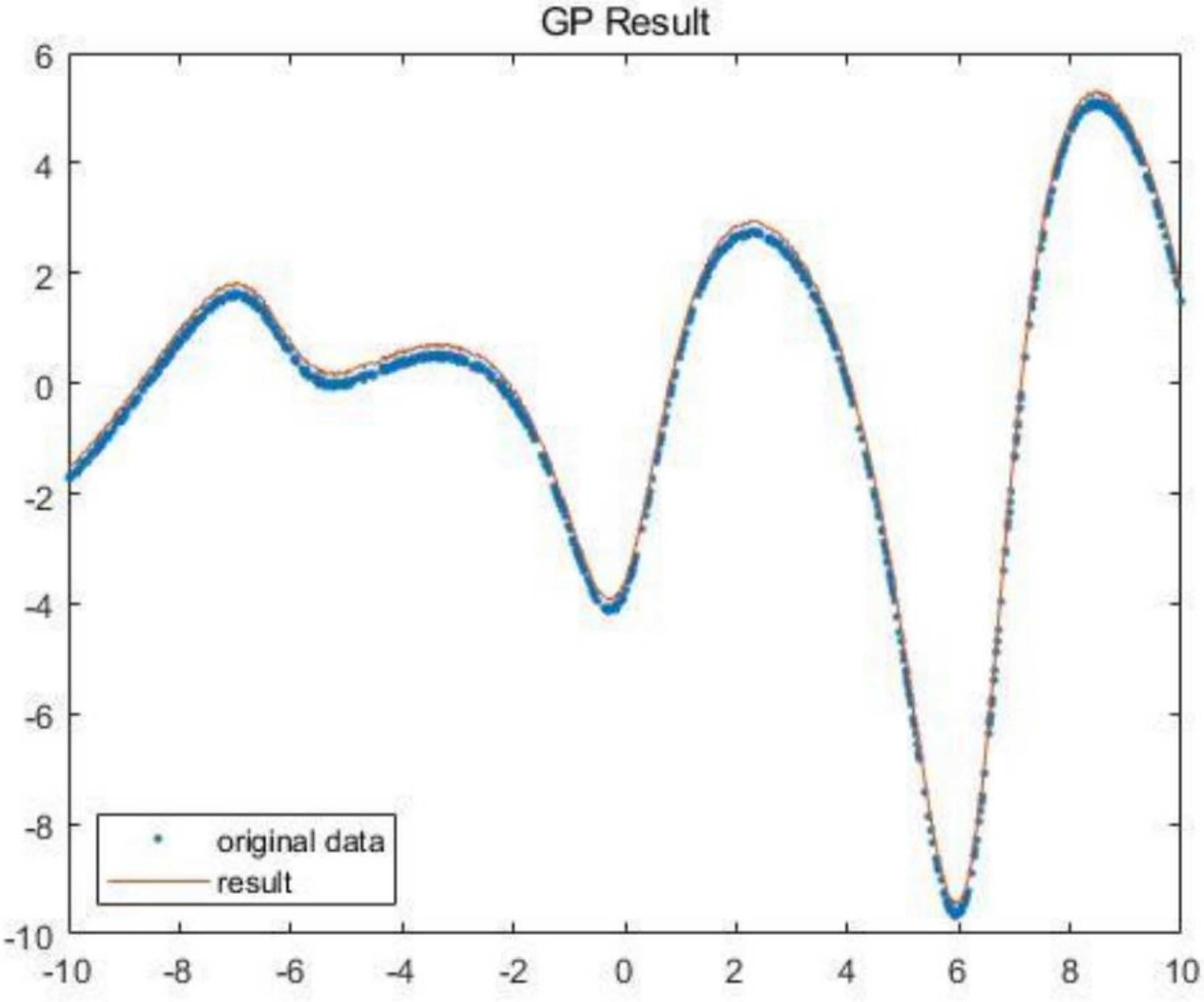
Seaborn

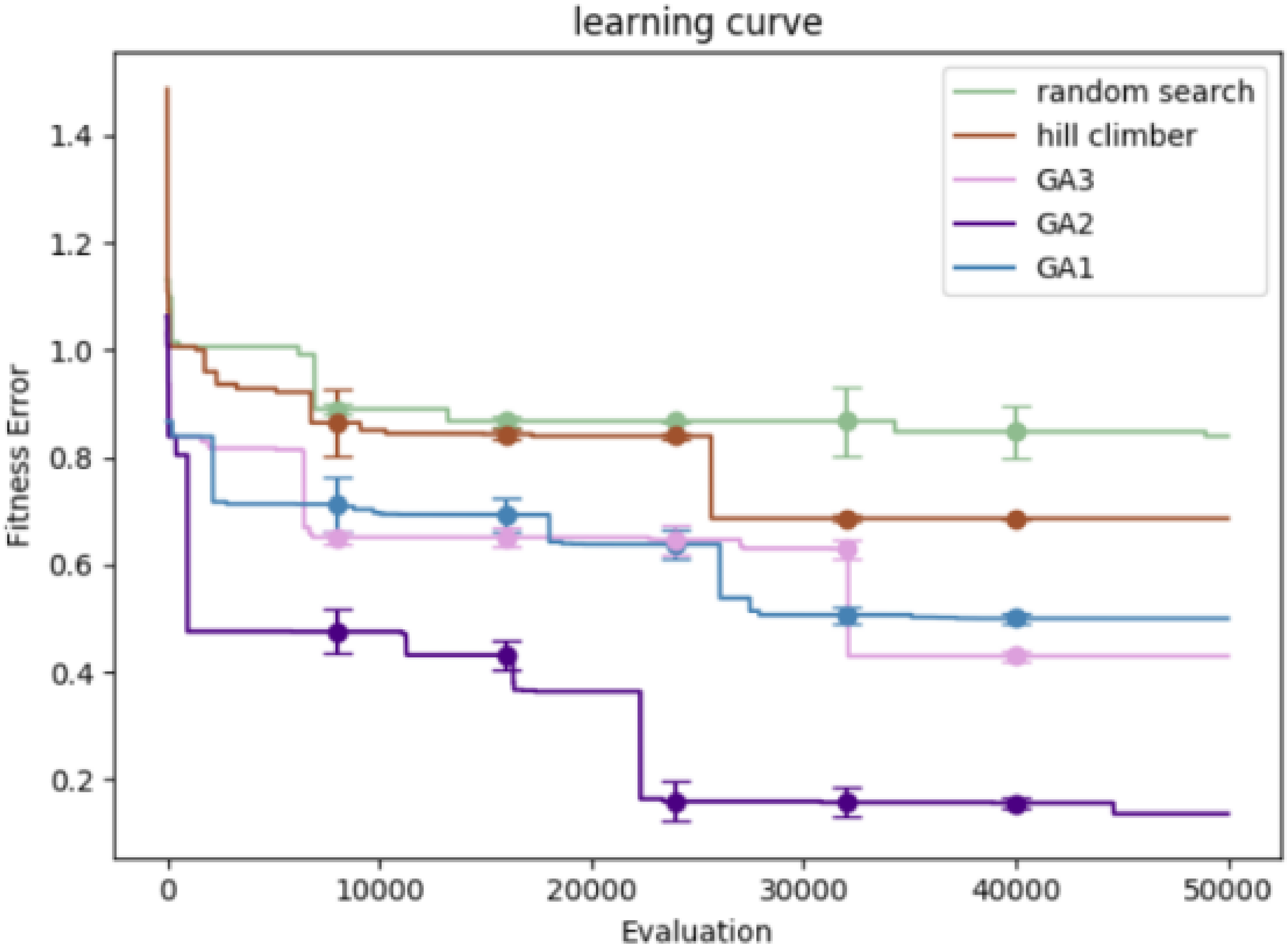


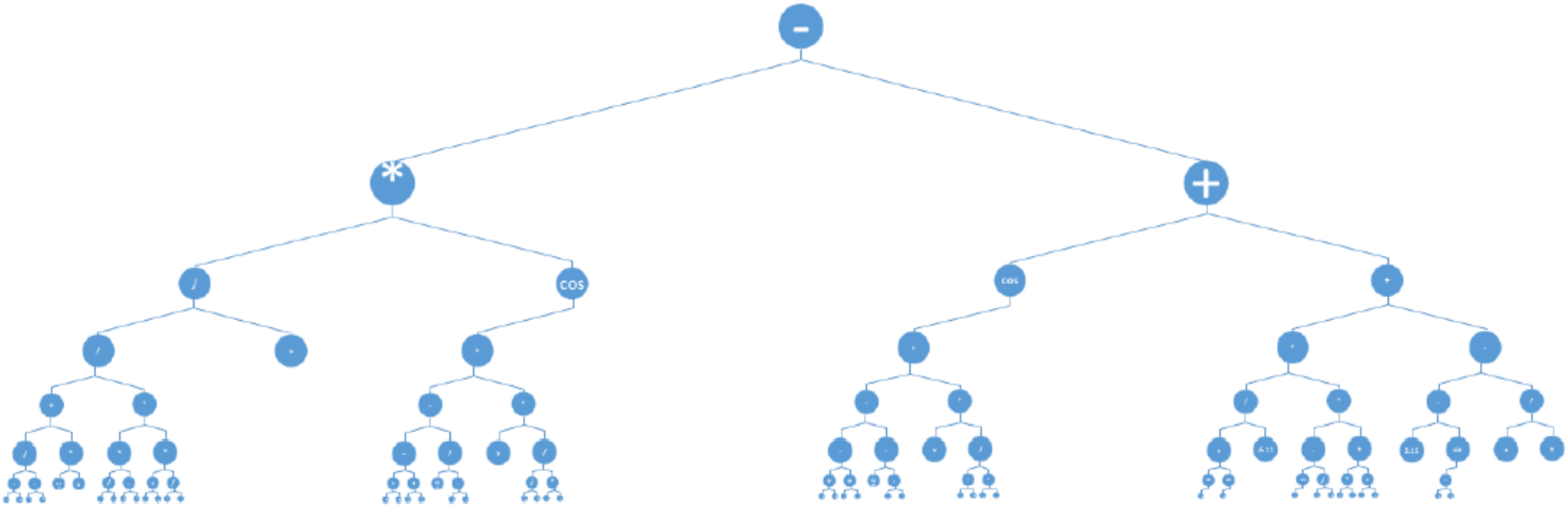
Boxi Xia

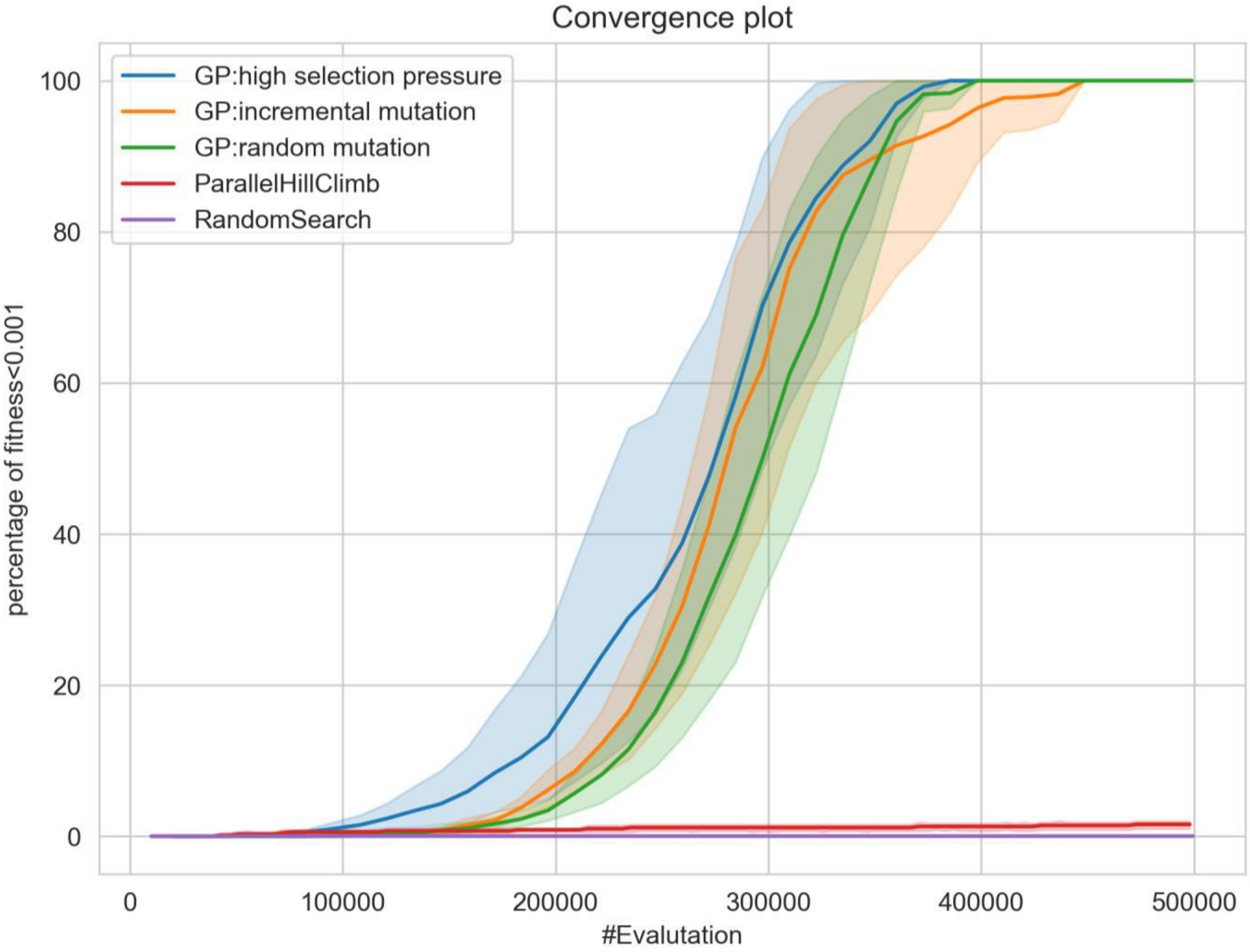

Parallel Hill Climber
Random Search
GP Tournament
GP DC
3×10^0
2×10^0
10^0
6×10^{-1}
MAE
0
2000
4000
6000
8000
10000
#Evaluations x 256


RMSE vs. Number of Evaluations
Best Random
Best HC
Best GP
3×10^1
2×10^1
10^1
RMSE
10^0
10^1
10^2
10^3
Number of Evaluations

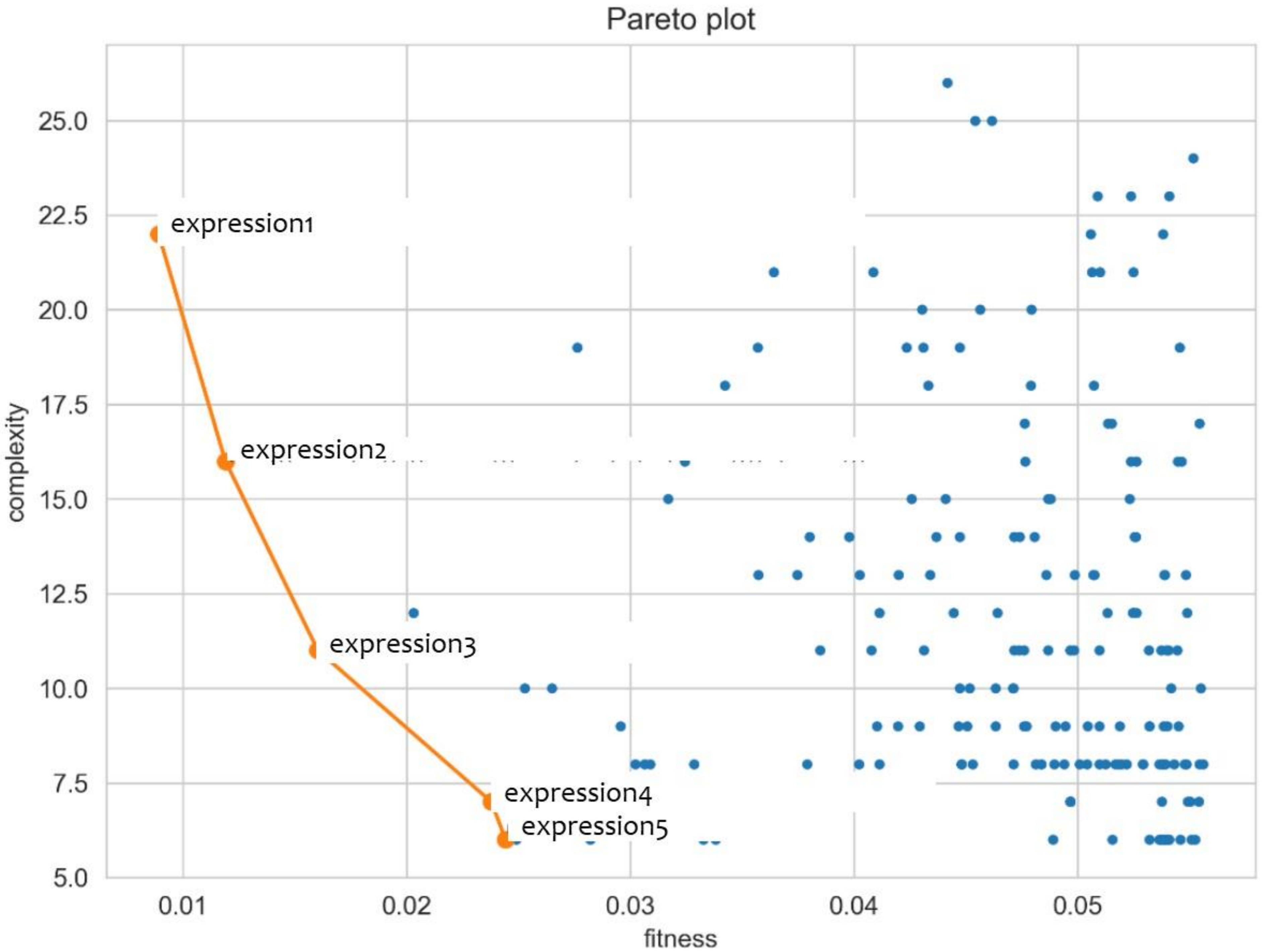

GP Result
6
4
2
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10
original data
result


learning curve
random search
hill climber
GA3
GA2
GA1
Fitness Error
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
10000
20000
30000
40000
50000
Evaluation


-
*
+
/
cos
cos
+

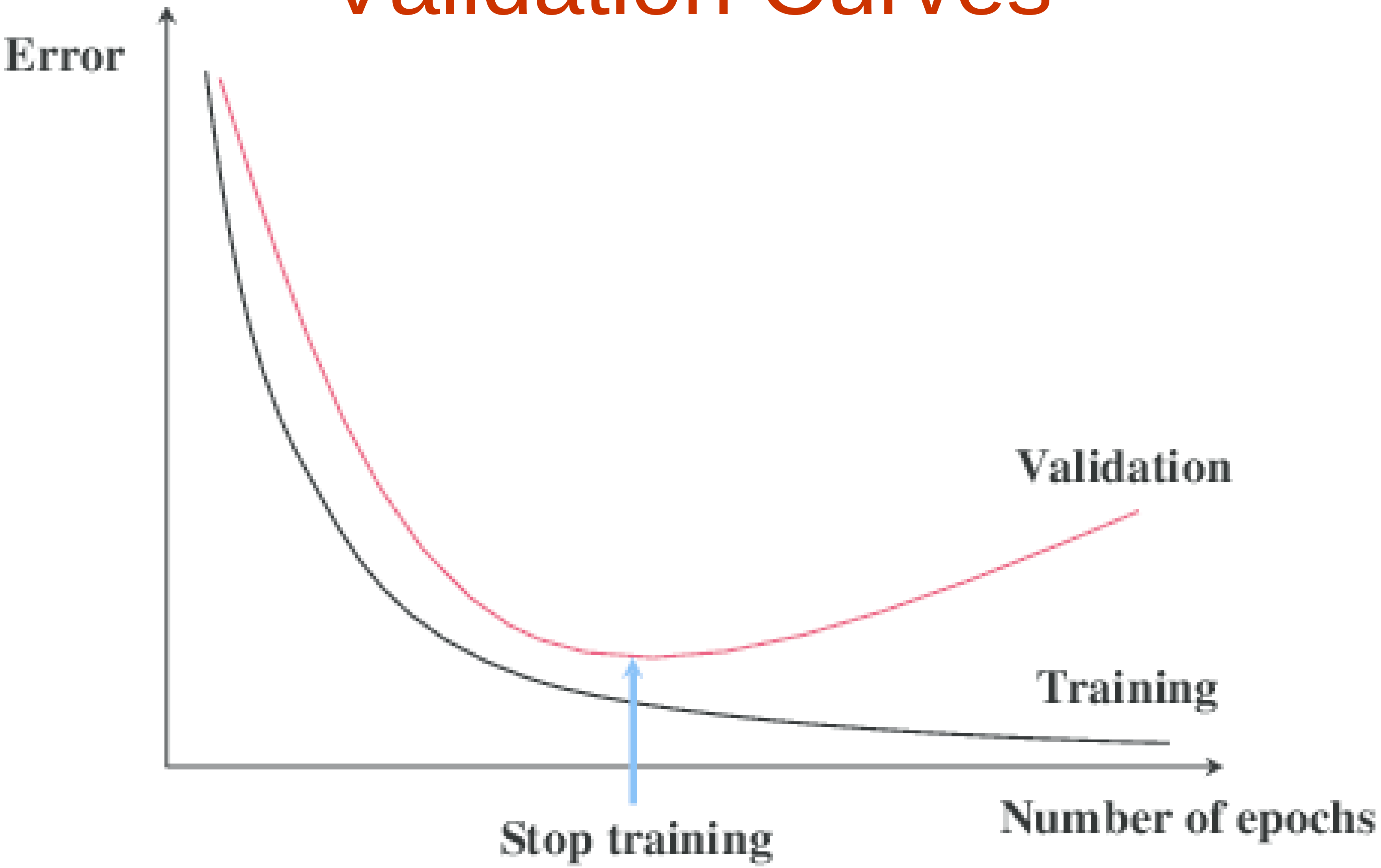

Convergence plot
GP:high selection pressure
GP:incremental mutation
GP:random mutation
ParallelHillClimb
RandomSearch
percentage of fitness<0.001
100
80
60
40
20
0
0
100000
200000
300000
400000
500000
#Evalutation

Boxi Xia

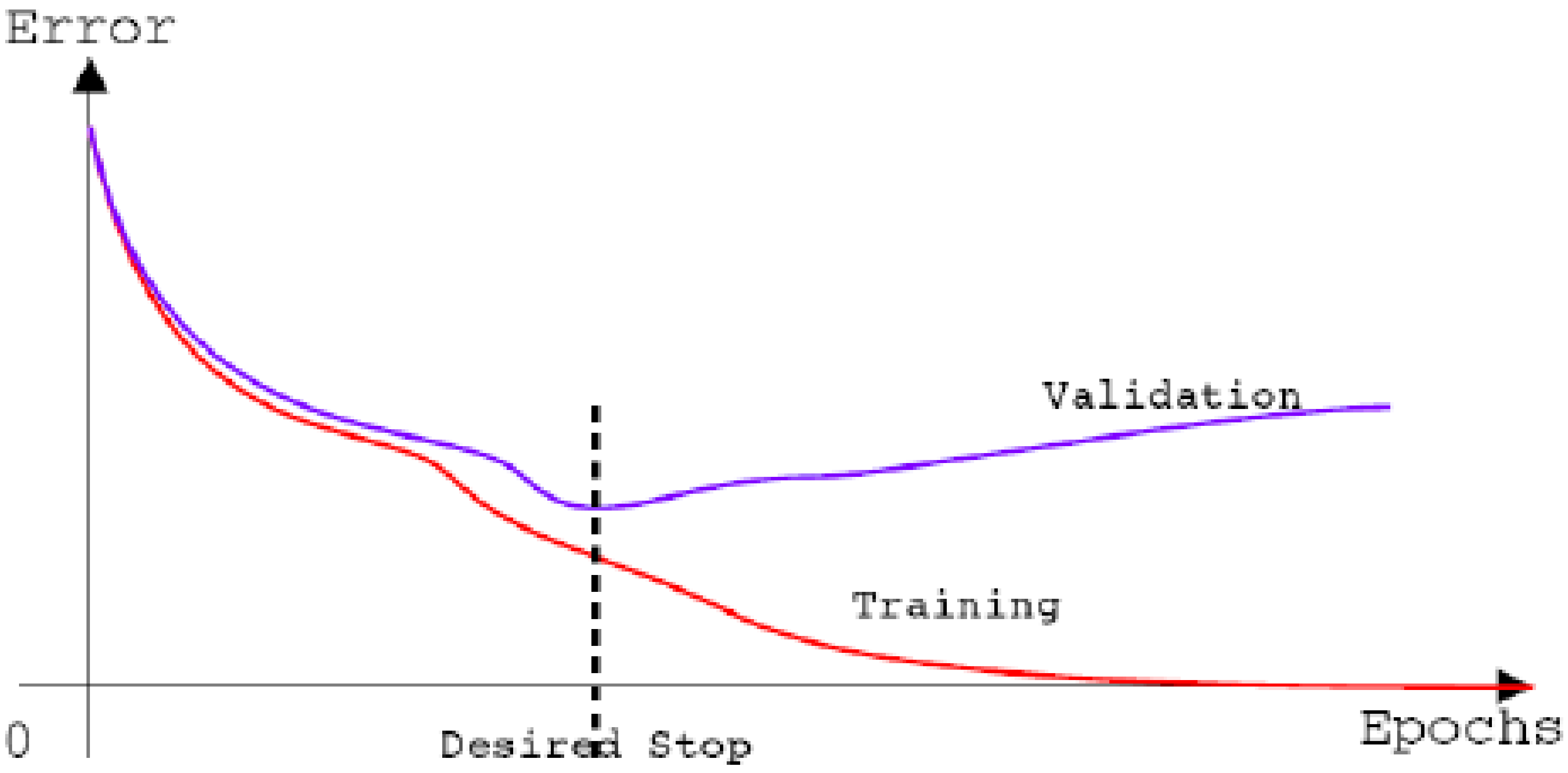

Pareto plot
complexity
fitness
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
0.01
0.02
0.03
0.04
0.05
expression1
expression2
expression3
expression4
expression5

Boxi Xia

Validation Curves

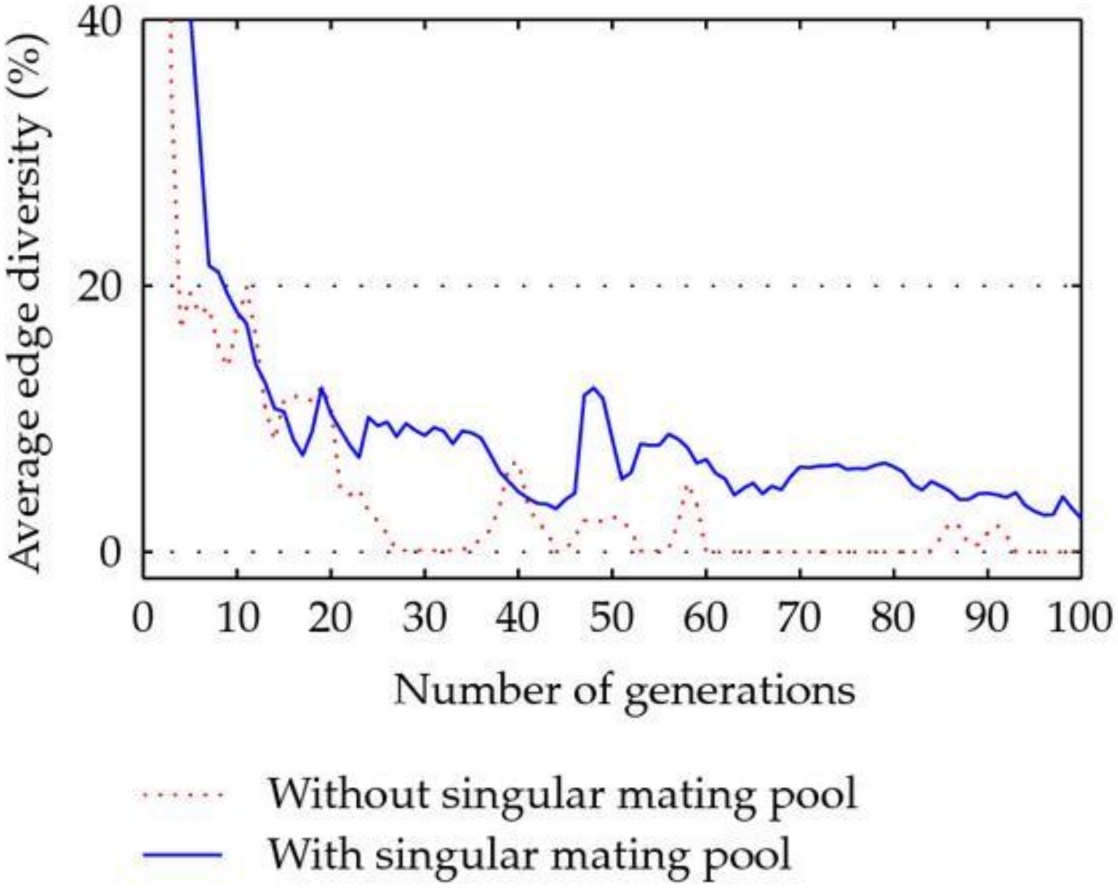


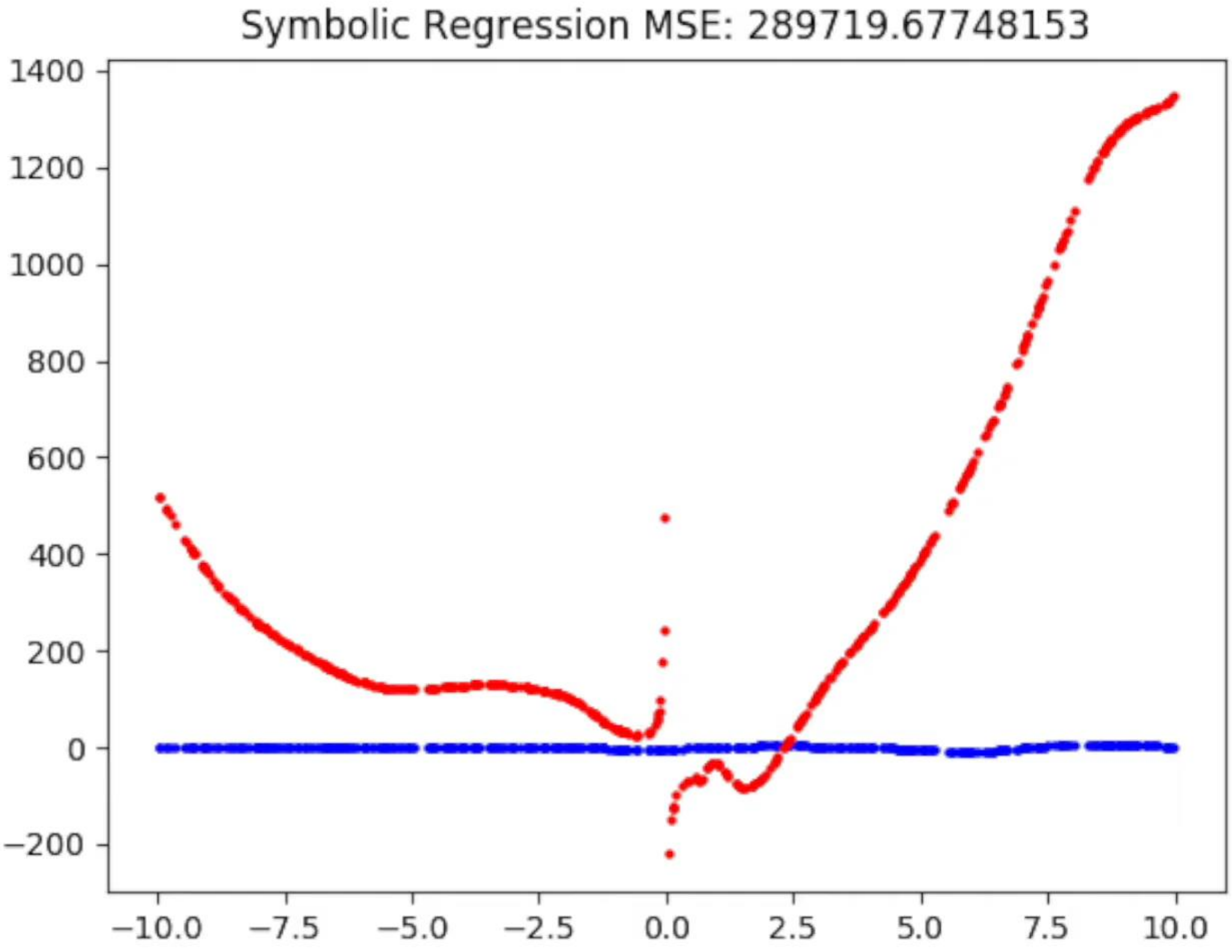
https://en.wikipedia.org/wiki/Early_stopping



https://en.wikipedia.org/wiki/Early_stopping

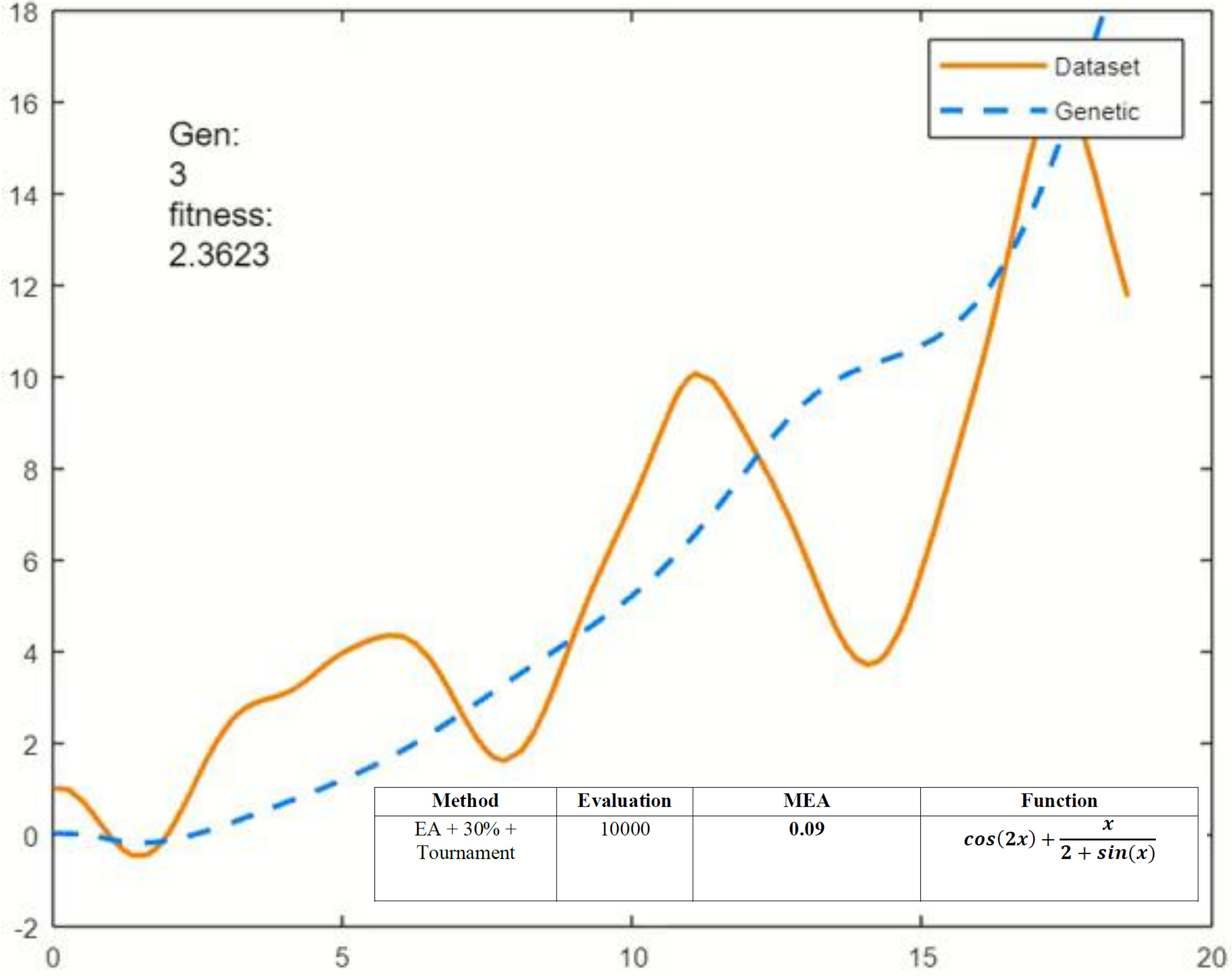
Diversity


Average edge diversity (%)
40
20
0
0
10
20
30
40
50
60
70
80
90
100
Number of generations
Without singular mating pool
With singular mating pool



Symbolic Regression MSE: 289719.67748153
1400
1200
1000
800
600
400
200
0
−200
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Max Segan

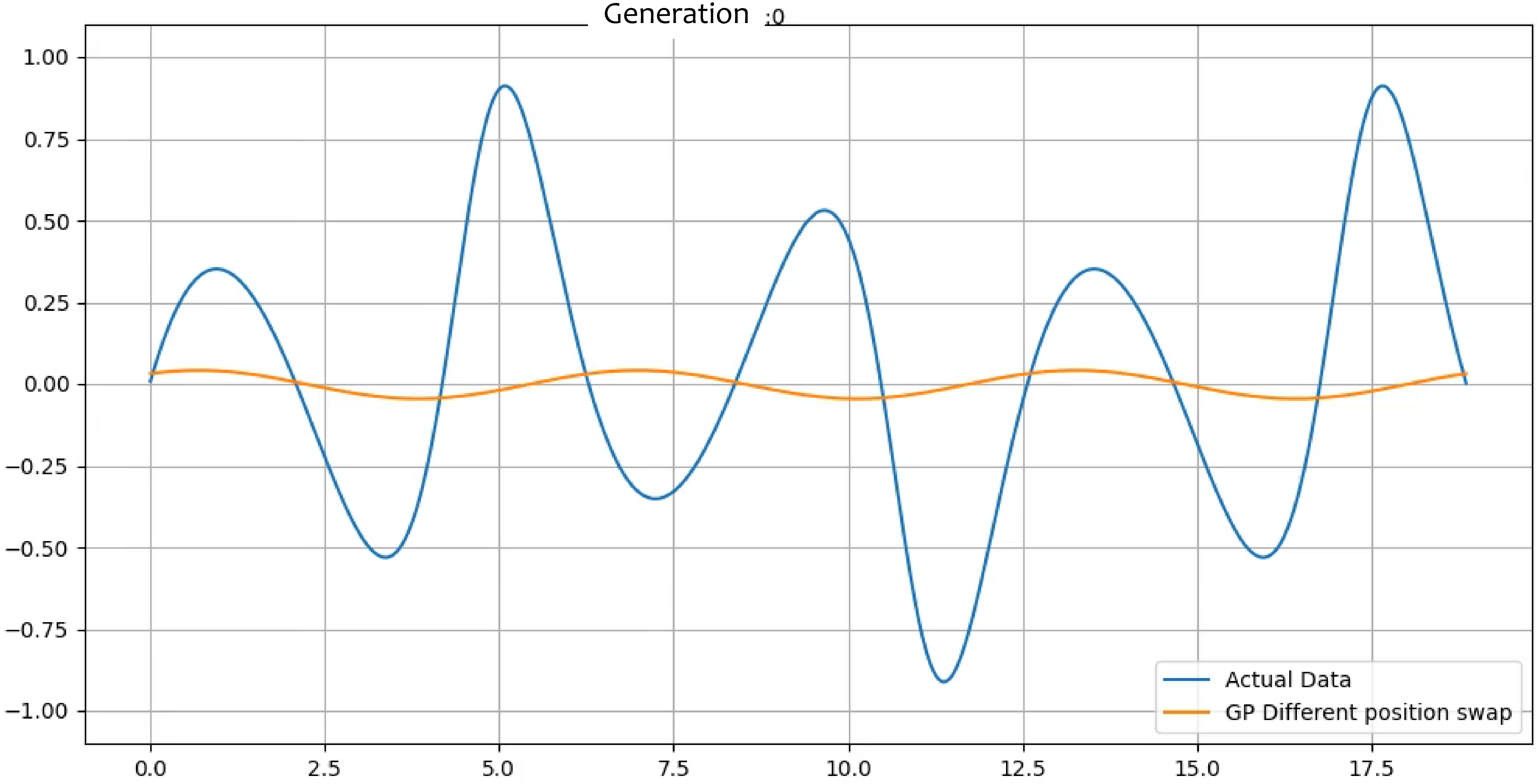


Method	Evaluation	MEA	Function
EA + 30% + Tournament	10000	**0.09**	$\boldsymbol{cos(2x)+\frac{x}{2+sin(x)}}$

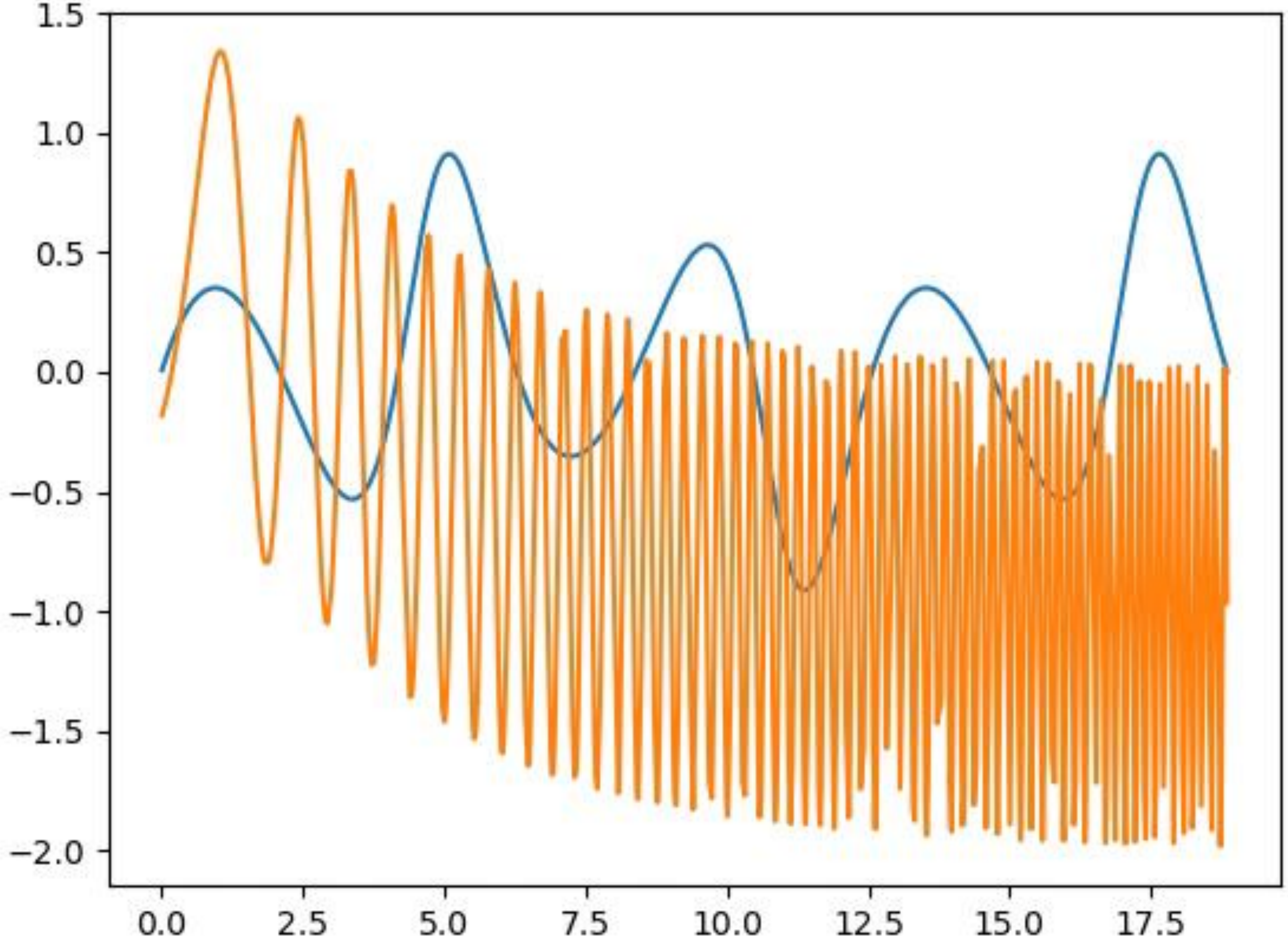
Jiong Lin

Pop size: 1000

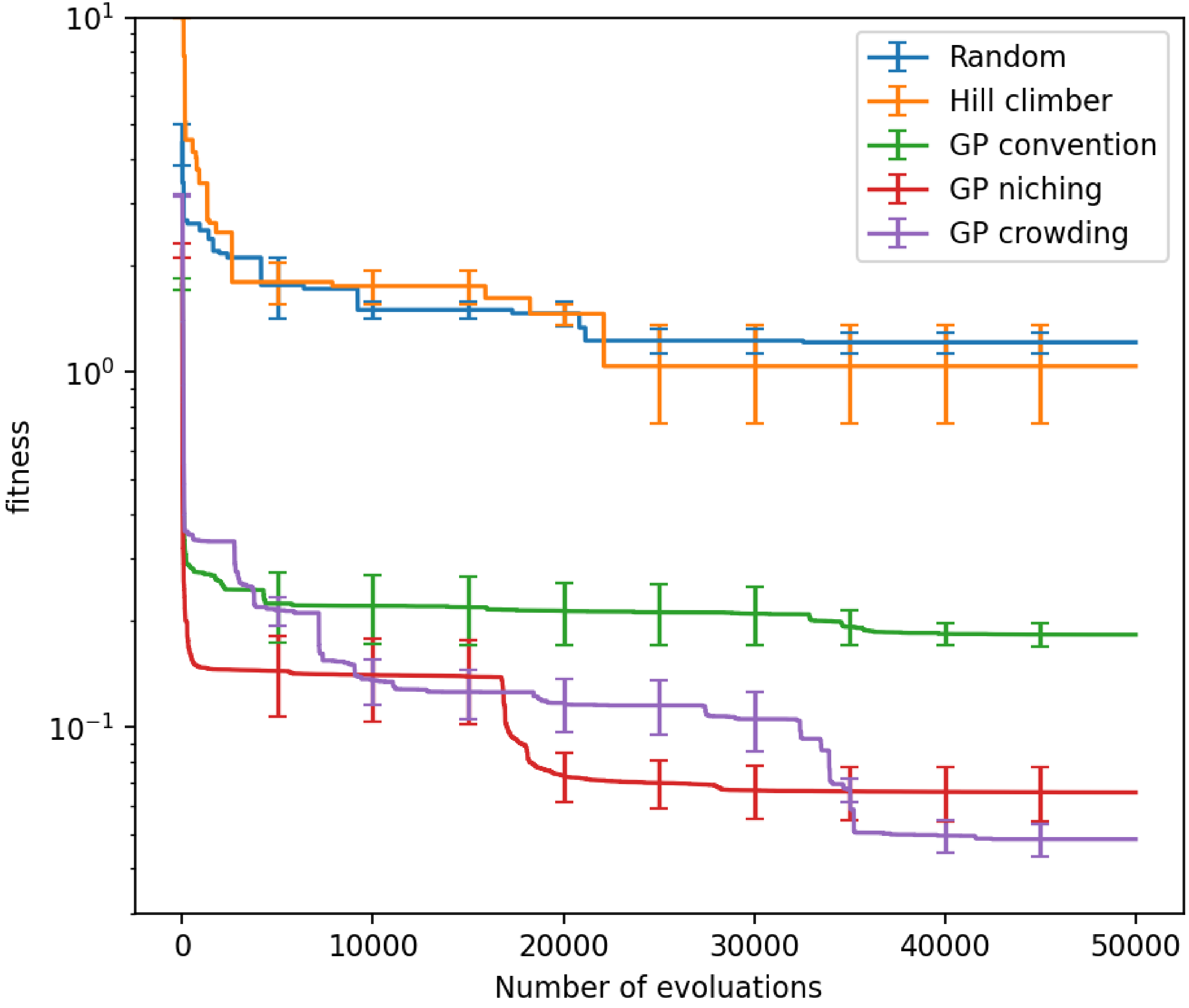
Generation :0



Mia Liu and Yushen Yang



Judah Goldfeder. John Schlaerth


10^1
10^0
10^{-1}
fitness
0
10000
20000
30000
40000
50000
Number of evoluations
Random
Hill climber
GP convention
GP niching
GP crowding

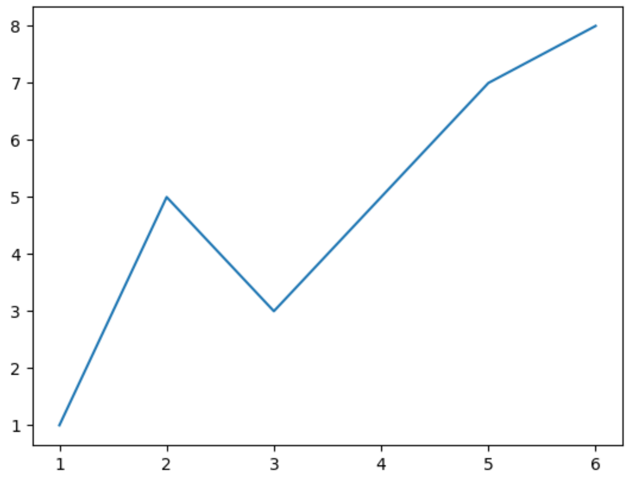
Jiong Lin

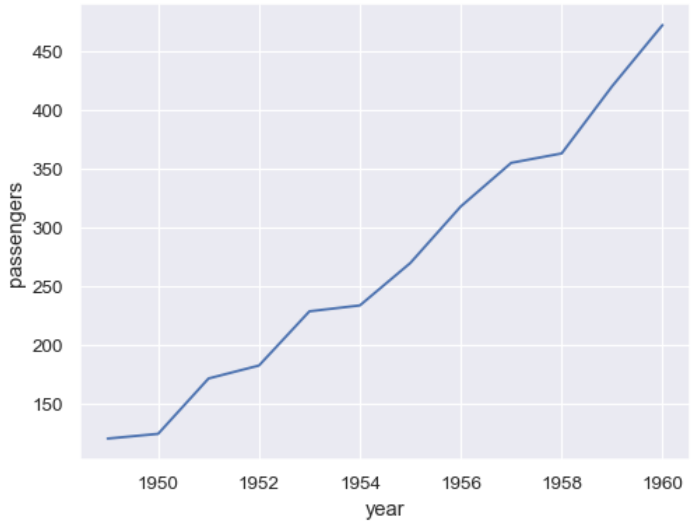
Tips

- Plot curves first. Don’t leave the curve-plotting to the end.
  - Learning curves can help you debug.
  - Always bring curves to office hours
- You can get max points even if you don’t solve the HW
  - Most of the grade is on process, not results
- Learn to use the cloud.
  - You can debug faster and work in parallel
- Develop the EA first (with crossover and mutation)
  - Disable the crossover and you have hill climber
  - Apply mutation to a blank solution and you have random search

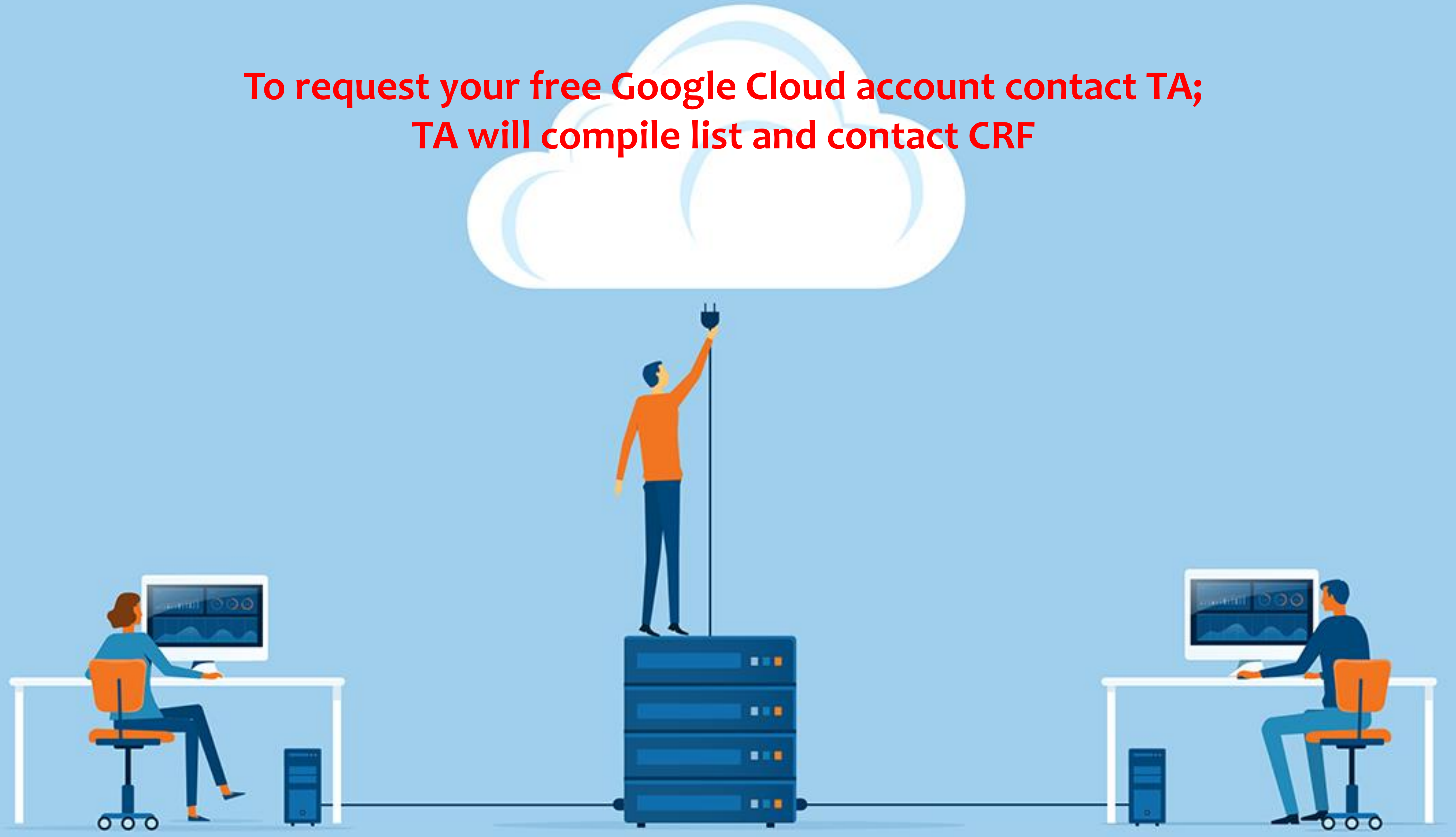
Charting in Python

- Matplotlib (https://matplotlib.org/)
  - Example: https://stackabuse.com/matplotlib-line-plot-tutorial-and-examples/
- Seaborn (https://seaborn.pydata.org/)
  - Example: https://seaborn.pydata.org/generated/seaborn.lineplot.html





To request your free Google Cloud account contact TA;
TA will compile list and contact CRF



GOOGLE CLOUD PLATFORM

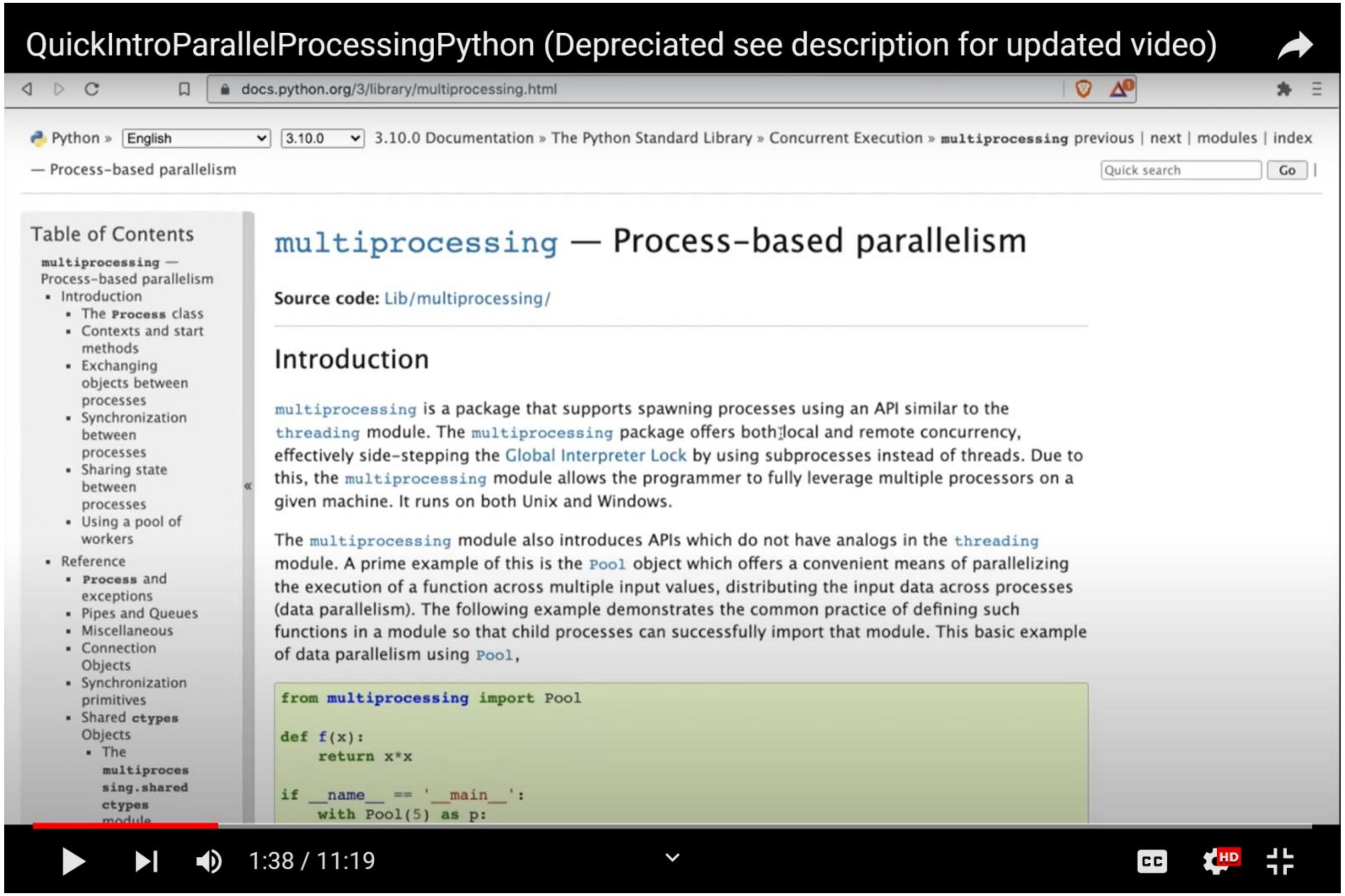
GETTING STARTED

BY PHILIPPE WYDER

https://youtu.be/cmLfMVorjgI

Philippe Wyder

Parallel Computing (Python/MATLAB)



QuickIntroParallelProcessingPython (Depreciated see description for updated video)

docs.python.org/3/library/multiprocessing.html

Python » English 3.10.0 3.10.0 Documentation » The Python Standard Library » Concurrent Execution » **multiprocessing** — Process-based parallelism | previous | next | modules | index

Quick search Go

multiprocessing — Process-based parallelism

Source code: Lib/multiprocessing/

Introduction

multiprocessing is a package that supports spawning processes using an API similar to the threading module. The multiprocessing package offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine. It runs on both Unix and Windows.

The multiprocessing module also introduces APIs which do not have analogs in the threading module. A prime example of this is the Pool object which offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism). The following example demonstrates the common practice of defining such functions in a module so that child processes can successfully import that module. This basic example of data parallelism using Pool,

```
from multiprocessing import Pool

def f(x):
    return x*x

if __name__ == '__main__':
    with Pool(5) as p:
```

PythonParallelComputingIntroduction - YouTube

Philippe Wyder