Assignment3：

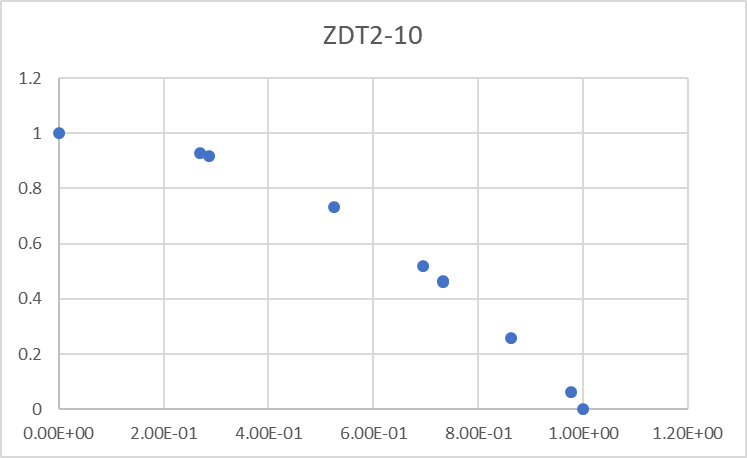
Yixuan Liu a1733854

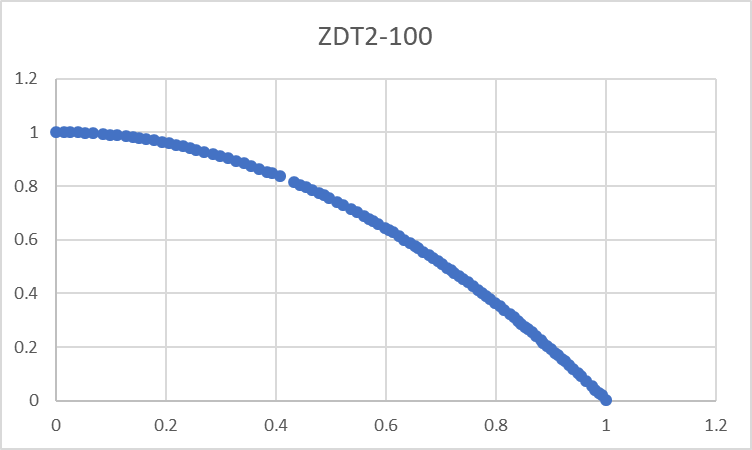
Contributions:

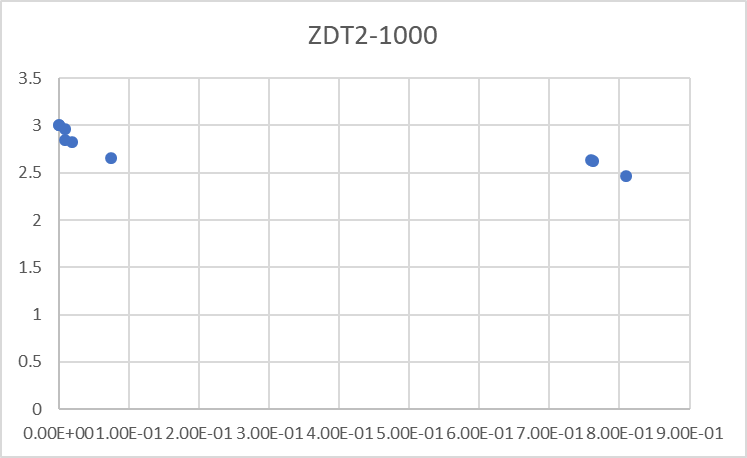
Yixuan Liu is responsible for every exercises’ coding and running

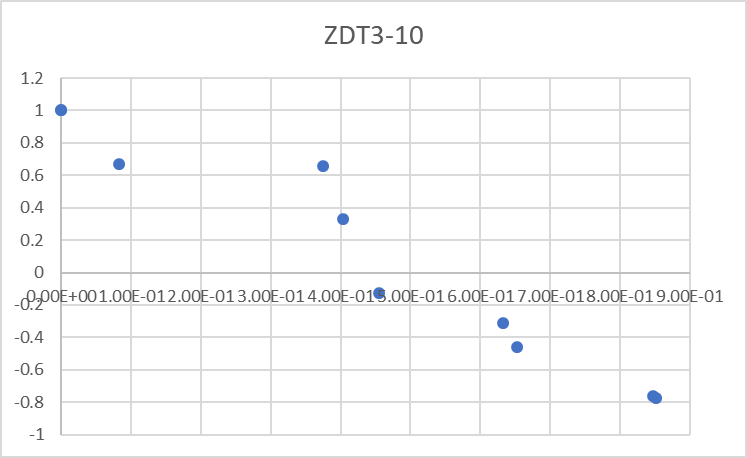
Exercise1:

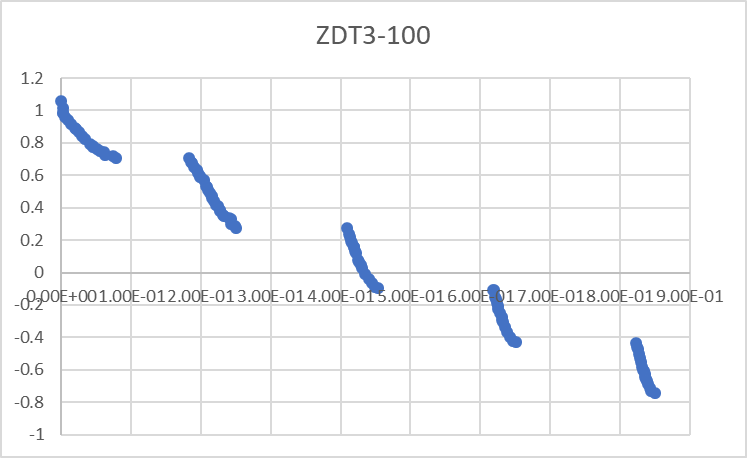
Result data are in folder “jMetal\jMetal-master”, as the same names as charts.

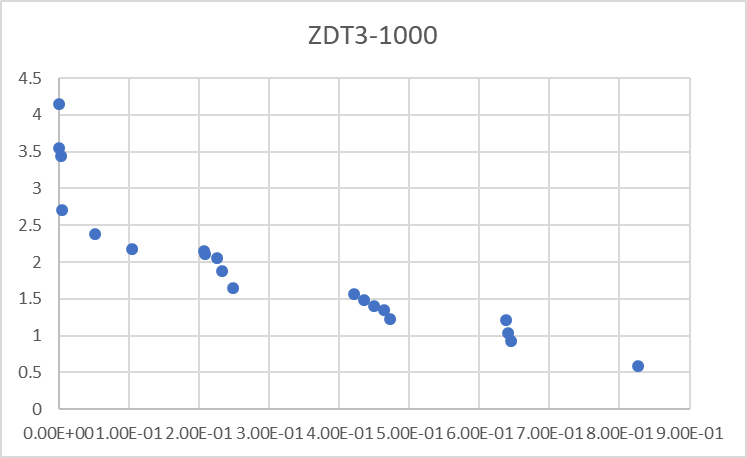












Exercise2:

To run the code of Exercise2, 3 and 4, the execution file for Exercise 2, 3, and 4 are in

“E:\jMetal\jMetal-master\jmetal-exec\src\main\java\org\uma\jmetal\runner\multiobjective”,

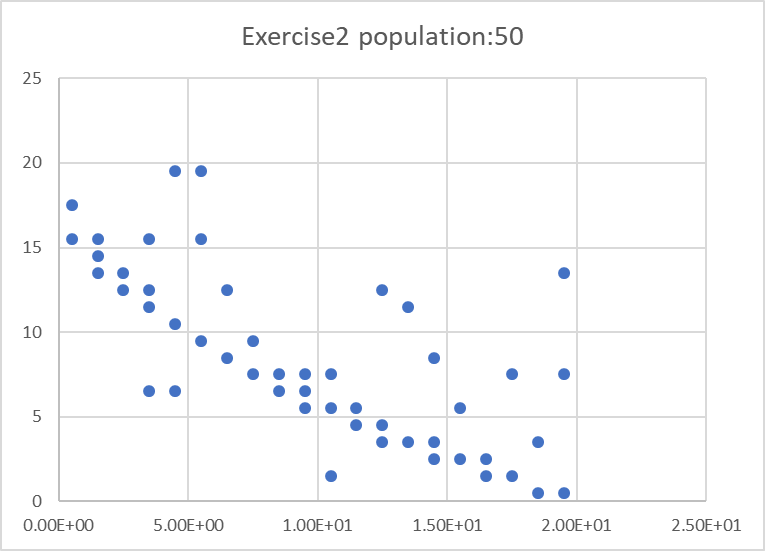
Exercise2exec.java is code for exercise2,

Exercise3exec.java is code for exercise3,

Exercise4exec.java is code for exercise4,

The implementation follows the requirement, the mutation operator randomly generate a number for variable values, and the crossover operator exchange partial variable values between two parents solutions. The operators are shared for three algorithms. The mutation operator is in “jMetal\jMetal-master\jmetal-core\src\main\java\org\uma\jmetal\operator\impl\mutation” , named Exercise2Mutation.java, the crossover operator is in “jMetal\jMetal-master\jmetal-core\src\main\java\org\uma\jmetal\operator\impl\crossover”, named ExerciseCrossover.java.

Algorithm A1 is the 2-Opt Algorithm, A2 is the Inverover Algorithm, and A3 is the best evolutionary algorithm from Assignment 1. The log file is Exercise2log.docx



Population: 100

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):10.864932708628885

J / K = Algorithm3, I = Algorithm2

p(i,j):-167.2862704000507

J / K = Algorithm1, I = Algorithm2

p(i,j):-167.84349530549207

J / K = Algorithm1, I = Algorithm3

p(i,j):9.570304200827763

J / K = Algorithm2, I = Algorithm1

p(i,j):197.0451532477467

J / K = Algorithm2, I = Algorithm3

p(i,j):192.48484873684345

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):12.405431360182092

J / K = Algorithm3, I = Algorithm2

p(i,j):-180.02479741037905

J / K = Algorithm1, I = Algorithm2

p(i,j):-178.41632916617962

J / K = Algorithm1, I = Algorithm3

p(i,j):6.501669203846944

J / K = Algorithm2, I = Algorithm1

p(i,j):206.98266397575728

J / K = Algorithm2, I = Algorithm3

p(i,j):194.83206094104375

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):9.577226062265254

J / K = Algorithm3, I = Algorithm2

p(i,j):-145.4312584247554

J / K = Algorithm1, I = Algorithm2

p(i,j):-148.35805923298267

J / K = Algorithm1, I = Algorithm3

p(i,j):7.237725315189522

J / K = Algorithm2, I = Algorithm1

p(i,j):175.87134531952452

J / K = Algorithm2, I = Algorithm3

p(i,j):171.5978077721057

Population: 50

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):2.9185732027587648

J / K = Algorithm3, I = Algorithm2

p(i,j):-16.084040461056574

J / K = Algorithm1, I = Algorithm2

p(i,j):-21.41478002506986

J / K = Algorithm1, I = Algorithm3

p(i,j):7.760794720956454

J / K = Algorithm2, I = Algorithm1

p(i,j):29.63755778810078

J / K = Algorithm2, I = Algorithm3

p(i,j):33.73633216896394

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):3.6558510674149858

J / K = Algorithm3, I = Algorithm2

p(i,j):-12.520721835201797

J / K = Algorithm1, I = Algorithm2

p(i,j):-19.20285787597831

J / K = Algorithm1, I = Algorithm3

p(i,j):6.487049924325319

J / K = Algorithm2, I = Algorithm1

p(i,j):30.633256136678625

J / K = Algorithm2, I = Algorithm3

p(i,j):31.253093711227294

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):3.816327499524121

J / K = Algorithm3, I = Algorithm2

p(i,j):-12.2935034221926

J / K = Algorithm1, I = Algorithm2

p(i,j):-17.539815521244606

J / K = Algorithm1, I = Algorithm3

p(i,j):4.573925136513594

J / K = Algorithm2, I = Algorithm1

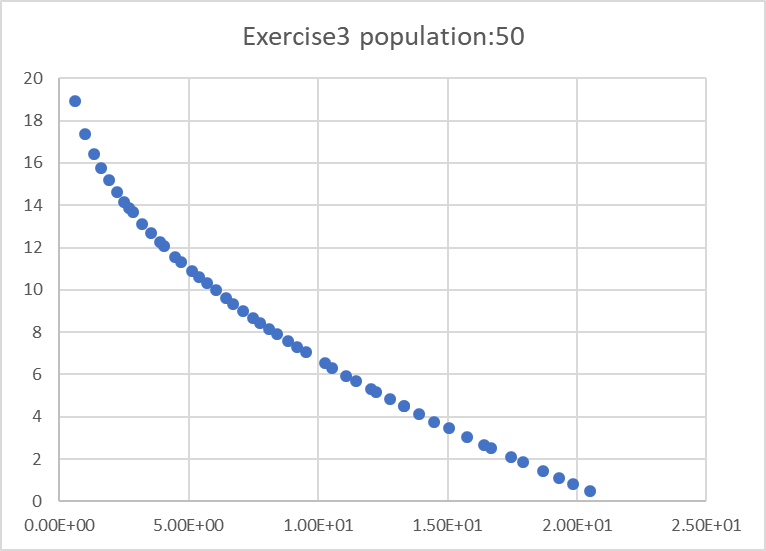
p(i,j):24.418830526019292

J / K = Algorithm2, I = Algorithm3

p(i,j):23.119102023191388

Exercise 3:

This implementation uses the same operators as Exercise2 but not restrict values to the central of the grid. Exercise3exec.java is code for exercise3. The log file is Exercise3log.docx



Population: 100

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):14.212499558156068

J / K = Algorithm3, I = Algorithm2

p(i,j):-128.77842826044815

J / K = Algorithm1, I = Algorithm2

p(i,j):-106.82619519044715

J / K = Algorithm1, I = Algorithm3

p(i,j):5.291853084473324

J / K = Algorithm2, I = Algorithm1

p(i,j):148.04999542583317

J / K = Algorithm2, I = Algorithm3

p(i,j):155.916661536265

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):12.572125993776552

J / K = Algorithm3, I = Algorithm2

p(i,j):-126.05565480613281

J / K = Algorithm1, I = Algorithm2

p(i,j):-125.9493685675349

J / K = Algorithm1, I = Algorithm3

p(i,j):6.78518636200279

J / K = Algorithm2, I = Algorithm1

p(i,j):153.46837715669807

J / K = Algorithm2, I = Algorithm3

p(i,j):147.60308050800057

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):0.9736744666005848

J / K = Algorithm3, I = Algorithm2

p(i,j):-107.35212946325632

J / K = Algorithm1, I = Algorithm2

p(i,j):-101.65172198981253

J / K = Algorithm1, I = Algorithm3

p(i,j):0.3328223121690428

J / K = Algorithm2, I = Algorithm1

p(i,j):121.45611080508367

J / K = Algorithm2, I = Algorithm3

p(i,j):125.75940806559679

Population: 50

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):0.7467152865257987

J / K = Algorithm3, I = Algorithm2

p(i,j):-14.912655456018243

J / K = Algorithm1, I = Algorithm2

p(i,j):-11.754767163973625

J / K = Algorithm1, I = Algorithm3

p(i,j):0.28930751506733543

J / K = Algorithm2, I = Algorithm1

p(i,j):23.989963818149576

J / K = Algorithm2, I = Algorithm3

p(i,j):25.588386589539667

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):1.4210854715202004E-14

J / K = Algorithm3, I = Algorithm2

p(i,j):-12.825039176458816

J / K = Algorithm1, I = Algorithm2

p(i,j):-17.79173433857939

J / K = Algorithm1, I = Algorithm3

p(i,j):1.4210854715202004E-14

J / K = Algorithm2, I = Algorithm1

p(i,j):27.927082638126123

J / K = Algorithm2, I = Algorithm3

p(i,j):28.431764727774024

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):7.105427357601002E-15

J / K = Algorithm3, I = Algorithm2

p(i,j):-6.428128425271865

J / K = Algorithm1, I = Algorithm2

p(i,j):-3.928810717102735

J / K = Algorithm1, I = Algorithm3

p(i,j):1.4210854715202004E-14

J / K = Algorithm2, I = Algorithm1

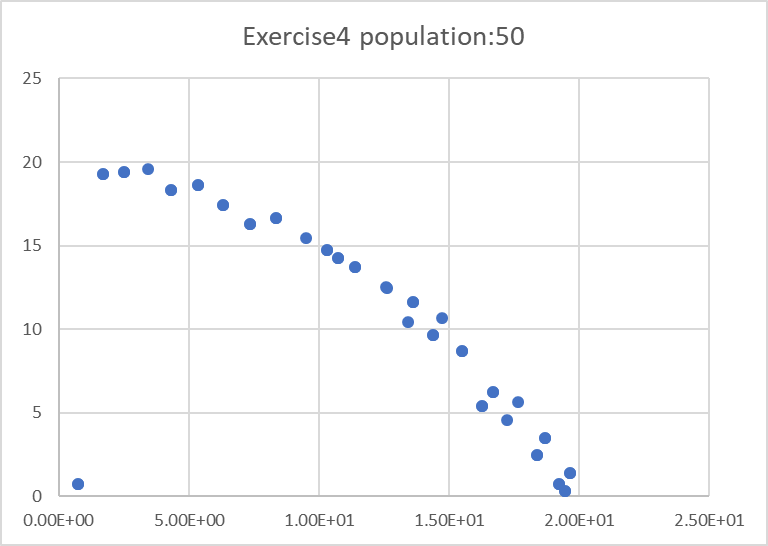
p(i,j):11.304163439252186

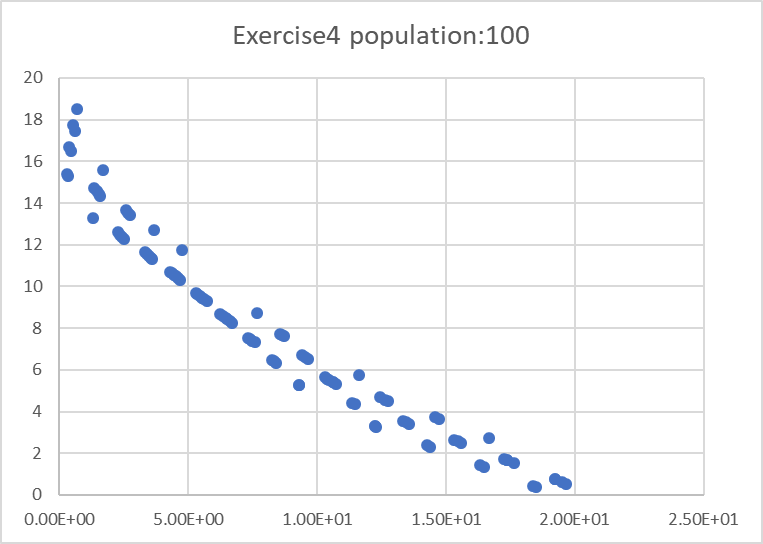
J / K = Algorithm2, I = Algorithm3

p(i,j):12.967501365549623

Exercise 4:

Exercise4exec.java is code for exercise4, this part is more interesting, firstly applying previous operators, record every node for each cell, after placing all nodes, apply a micro operator on each cell, all nodes have to be in the same cell after operations. The micro operator is the function doAMicro() in SPEARunner.java. The log file is Exercise4log.docx





Population: 100

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):11.454484925944755

J / K = Algorithm3, I = Algorithm2

p(i,j):-128.5574393578037

J / K = Algorithm1, I = Algorithm2

p(i,j):-121.28413398326983

J / K = Algorithm1, I = Algorithm3

p(i,j):-0.7267774441168058

J / K = Algorithm2, I = Algorithm1

p(i,j):151.1276677958747

J / K = Algorithm2, I = Algorithm3

p(i,j):142.41074553463437

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):6.347154217309821

J / K = Algorithm3, I = Algorithm2

p(i,j):-114.15390231007345

J / K = Algorithm1, I = Algorithm2

p(i,j):-121.60742102795767

J / K = Algorithm1, I = Algorithm3

p(i,j):3.25075441254279

J / K = Algorithm2, I = Algorithm1

p(i,j):150.36380159042756

J / K = Algorithm2, I = Algorithm3

p(i,j):144.93066326002122

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):1.9422750164917915

J / K = Algorithm3, I = Algorithm2

p(i,j):-101.61067552519636

J / K = Algorithm1, I = Algorithm2

p(i,j):-103.89666131746156

J / K = Algorithm1, I = Algorithm3

p(i,j):0.31617136767366105

J / K = Algorithm2, I = Algorithm1

p(i,j):115.96156189876861

J / K = Algorithm2, I = Algorithm3

p(i,j):113.53933797960983

Population: 50

NSGAII

J / K = Algorithm3, I = Algorithm1

p(i,j):1.4210854715202004E-14

J / K = Algorithm3, I = Algorithm2

p(i,j):-9.882626346701741

J / K = Algorithm1, I = Algorithm2

p(i,j):-14.592662759531137

J / K = Algorithm1, I = Algorithm3

p(i,j):0.9196520088354987

J / K = Algorithm2, I = Algorithm1

p(i,j):20.940387528573027

J / K = Algorithm2, I = Algorithm3

p(i,j):26.50554496087996

SPEA2

J / K = Algorithm3, I = Algorithm1

p(i,j):2.8421709430404007E-14

J / K = Algorithm3, I = Algorithm2

p(i,j):-15.257896960972559

J / K = Algorithm1, I = Algorithm2

p(i,j):-18.102507373502974

J / K = Algorithm1, I = Algorithm3

p(i,j):1.4210854715202004E-14

J / K = Algorithm2, I = Algorithm1

p(i,j):26.160485576553427

J / K = Algorithm2, I = Algorithm3

p(i,j):23.40084044749362

IBEA

J / K = Algorithm3, I = Algorithm1

p(i,j):0.01579142887194962

J / K = Algorithm3, I = Algorithm2

p(i,j):-11.240772425125272

J / K = Algorithm1, I = Algorithm2

p(i,j):-13.464151083715173

J / K = Algorithm1, I = Algorithm3

p(i,j):0.01579142887194962

J / K = Algorithm2, I = Algorithm1

p(i,j):19.55663228360926

J / K = Algorithm2, I = Algorithm3

p(i,j):18.163881468627608