

Further development of Particle Swarm Optimization method applied to accelerators

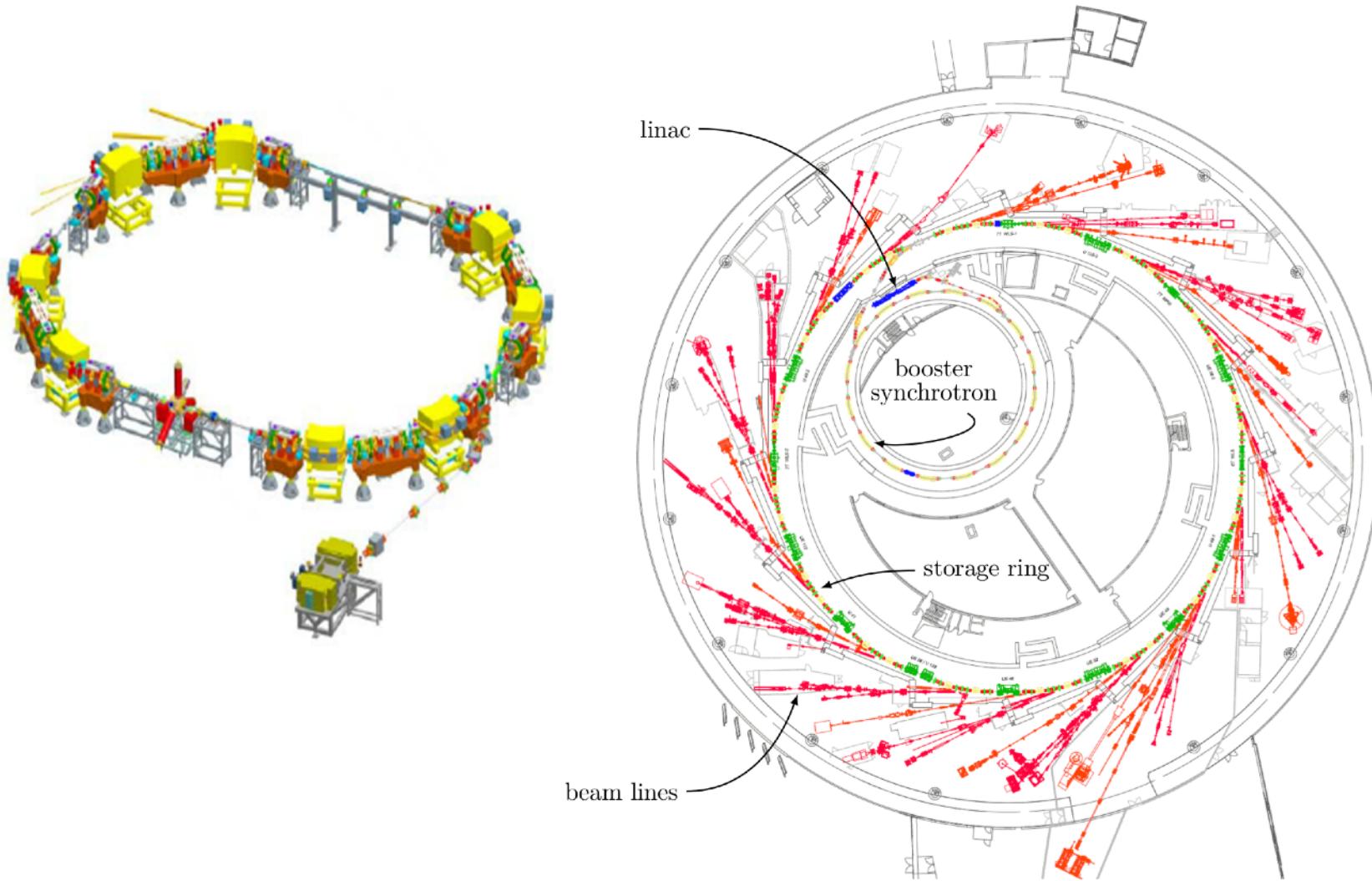
Illyas Fatkhullin

Supervisor: Dr. Ji Li

FG-IA

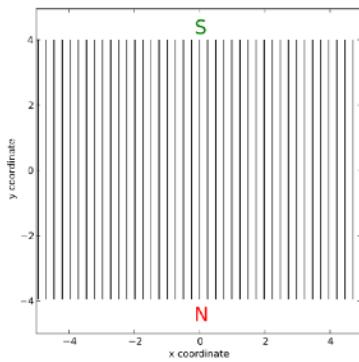
Helmholtz Zentrum Berlin, Germany

- Background and motivation
- Introduction of Particle Swarm Optimization (PSO)
- Improvements applied to PSO
- Experimental and simulation results
- Multi-objective PSO

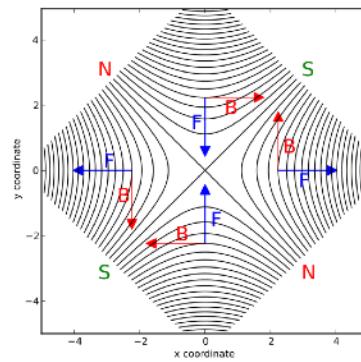


Field lines of an idealized
dipole, quadrupole and sextupole magnet
in the plane transverse to the beam direction

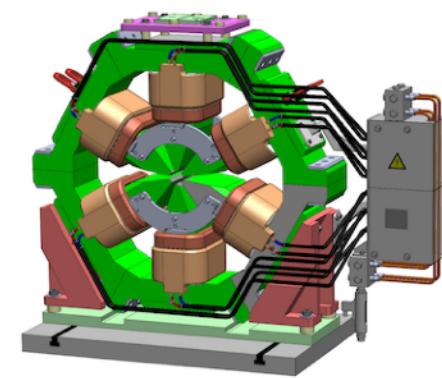
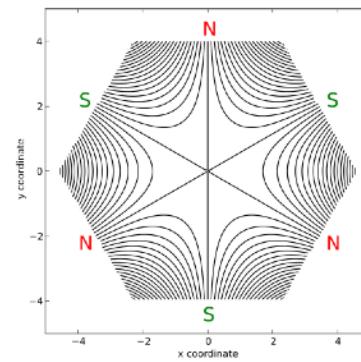
dipole



quadrupole



sextupole



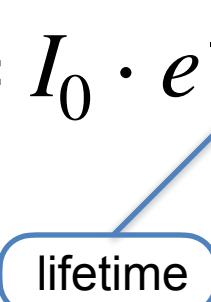
1)

Injection efficiency =

$$\frac{\text{Beam current injected to storage ring}}{\text{Beam current from injector}}$$

2)

$$I = I_0 \cdot e^{-\frac{t}{\tau}}$$


 lifetime

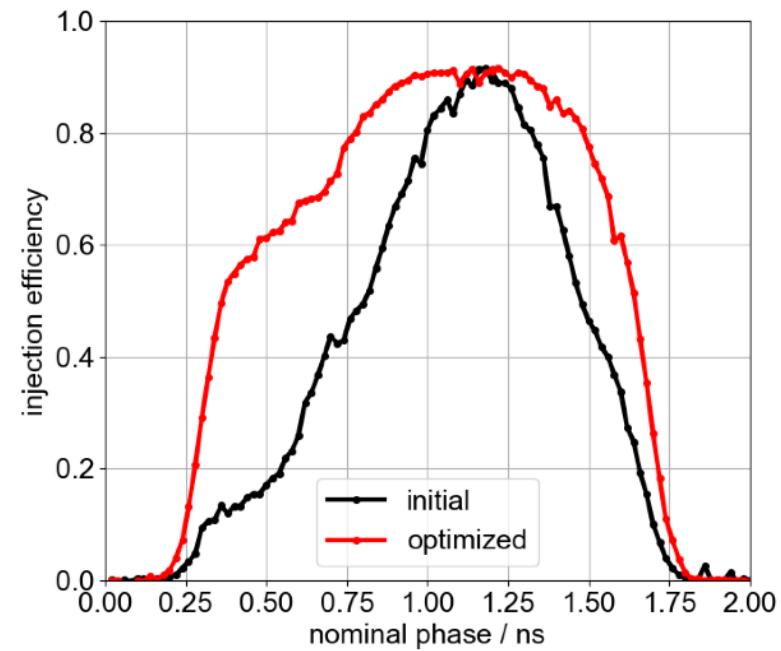
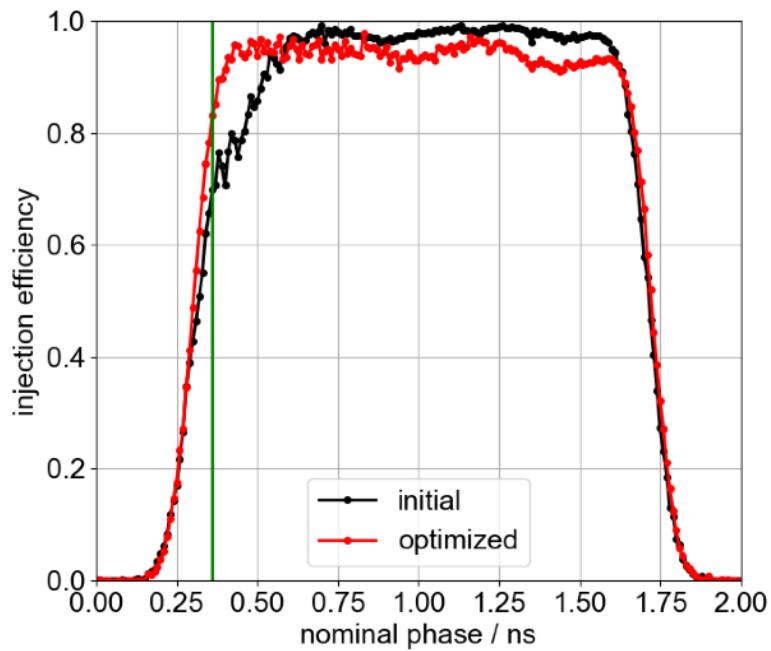
ring-Sextadl

Girag 2018/06/30

Ring Sextupoles

Cmds: []

Name	S	Setpoint	Readback	Mem	C
S1PR	[]	88.6272	88.578	88.6272	[]
S2PDR	[]	72.6533	72.640	72.6533	[]
S2PTR	[]	72.6533	72.639	72.6533	[]
S3PDR	[]	39.9495	39.911	39.9495	[]
S3PTR	[]	39.9495	39.934	39.9495	[]
S4PDR	[]	31.1680	31.121	31.1680	[]
S4PTR	[]	31.1680	31.119	31.1680	[]
S3PD1R	[]	39.9495	40.13	39.9495	[]
S4PD1R	[]	31.1680	31.45	31.1680	[]
S3P1T6R	[]	39.9495	40.20	39.9495	[]
S3P2T6R	[]	39.9495	39.97	39.9495	[]
S4P1T6R	[]	31.1680	31.30	31.1680	[]
S4P2T6R	[]	31.1680	31.24	31.1680	[]



- optimization of 10 sextupoles
- insertion devices deteriorate injection efficiency

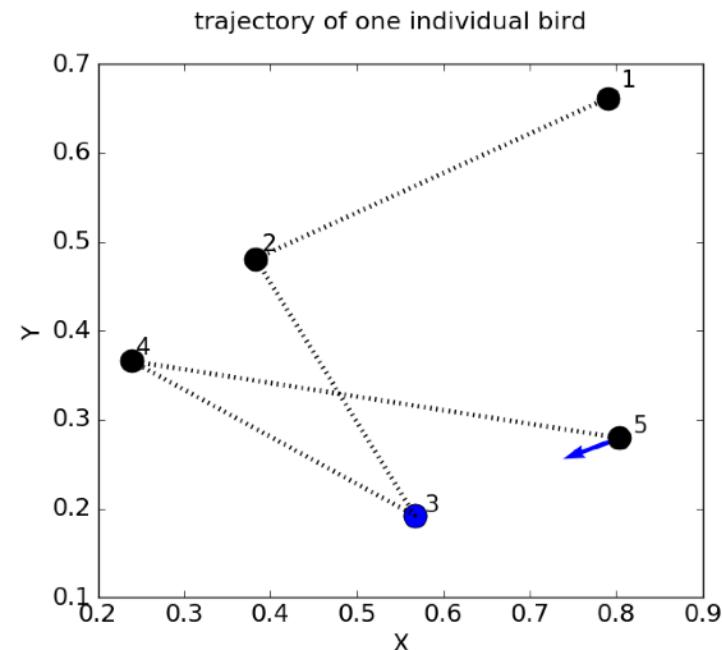
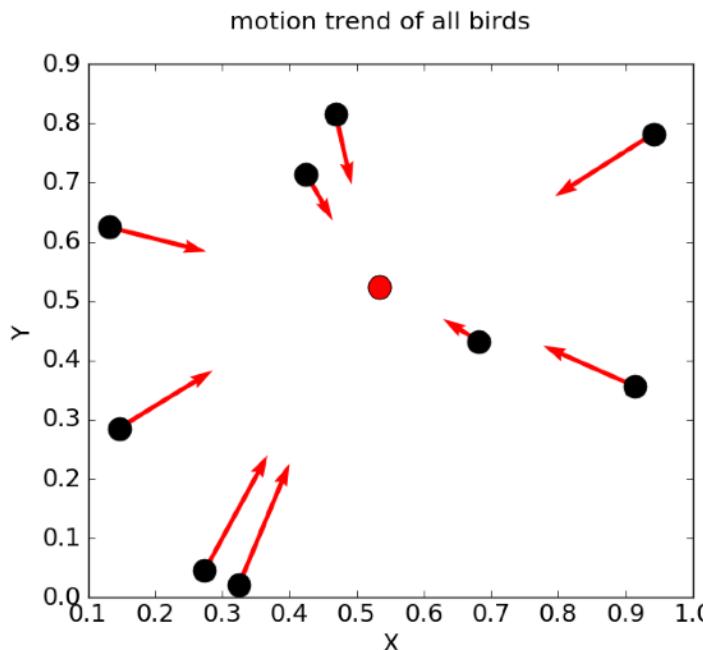


Birds in a flock searching for food

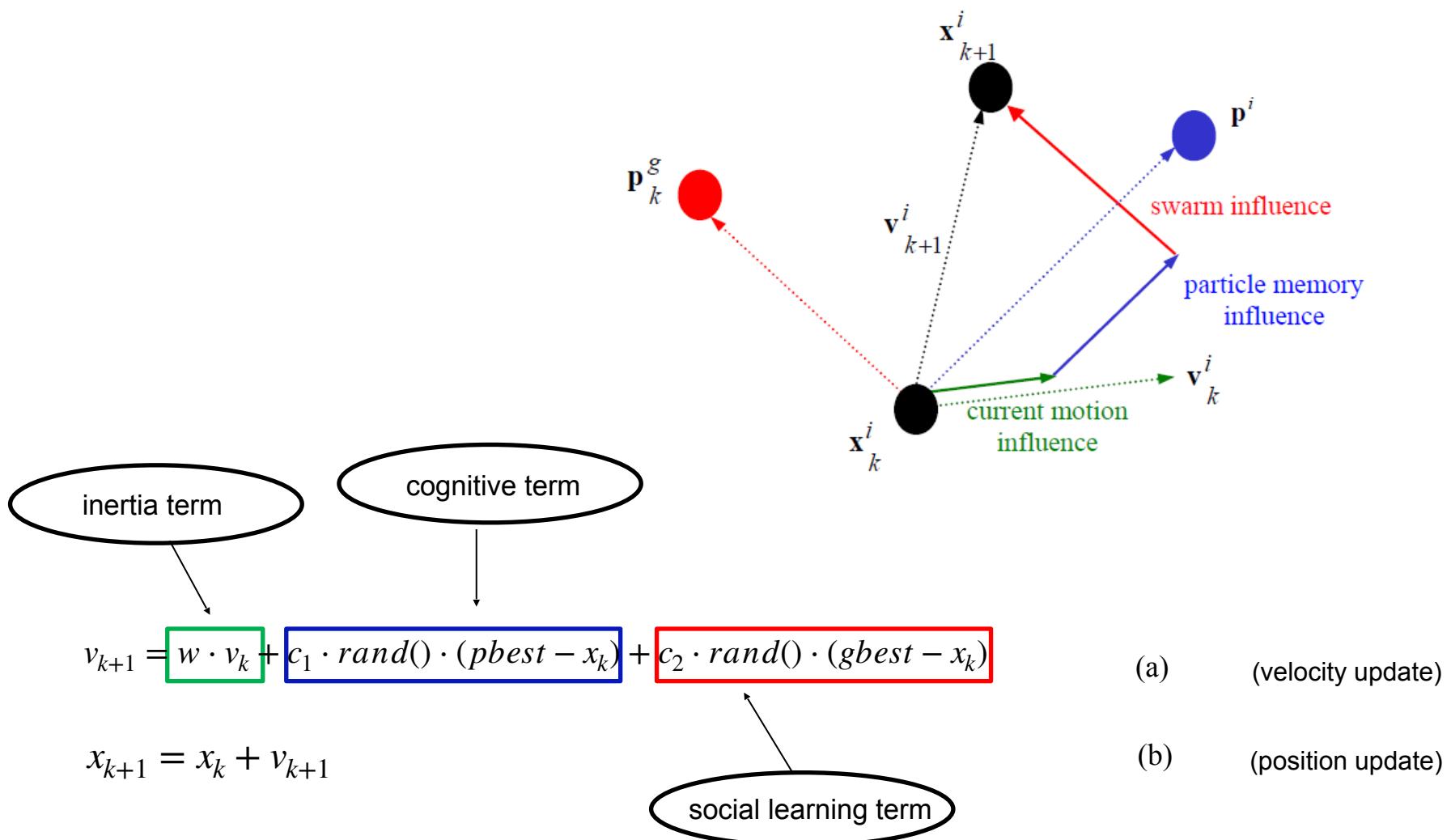
Developed by Dr. Eberhart and Dr. Kennedy in 1995

SCENARIO:

- A group of flying **black** birds searching the only one piece of food in a certain space
- No bird knows where the food is
- The bird closest to the food turns **red** and can be recognized (**gbest**)
- The birds have memory, every bird marks the best position in its flightpath with **blue** color (**pbest**)
- All birds follow **the red bird** and are influenced by their **blue positions** until the food is found



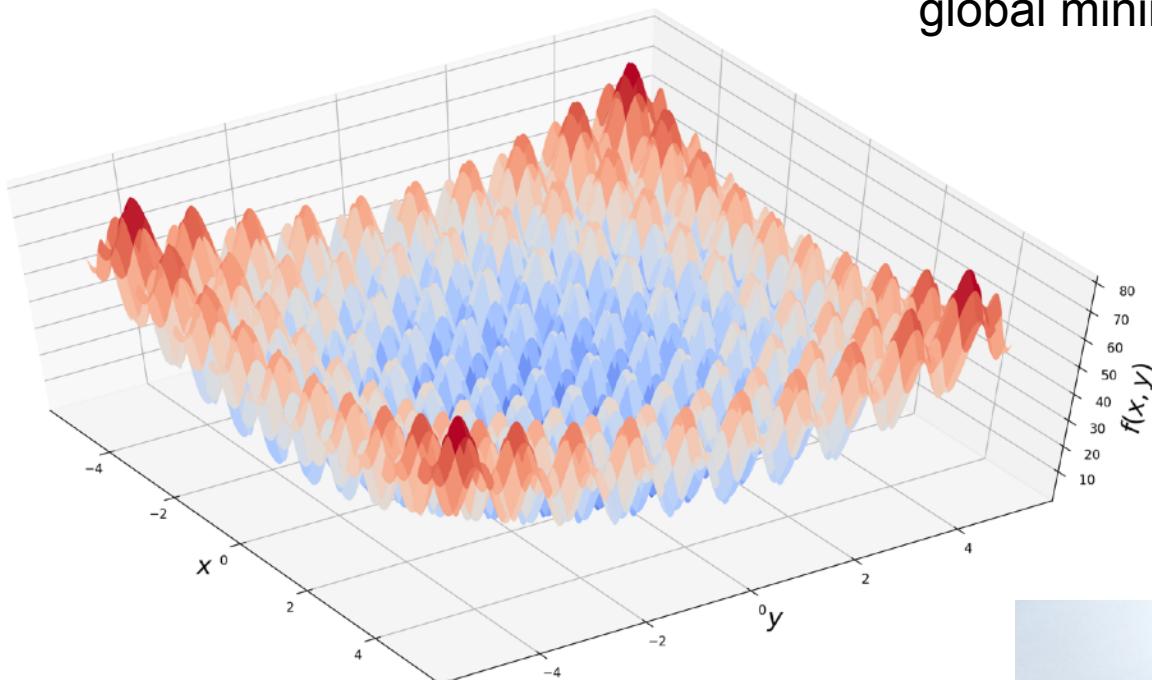
MAIN FORMULA OF PSO

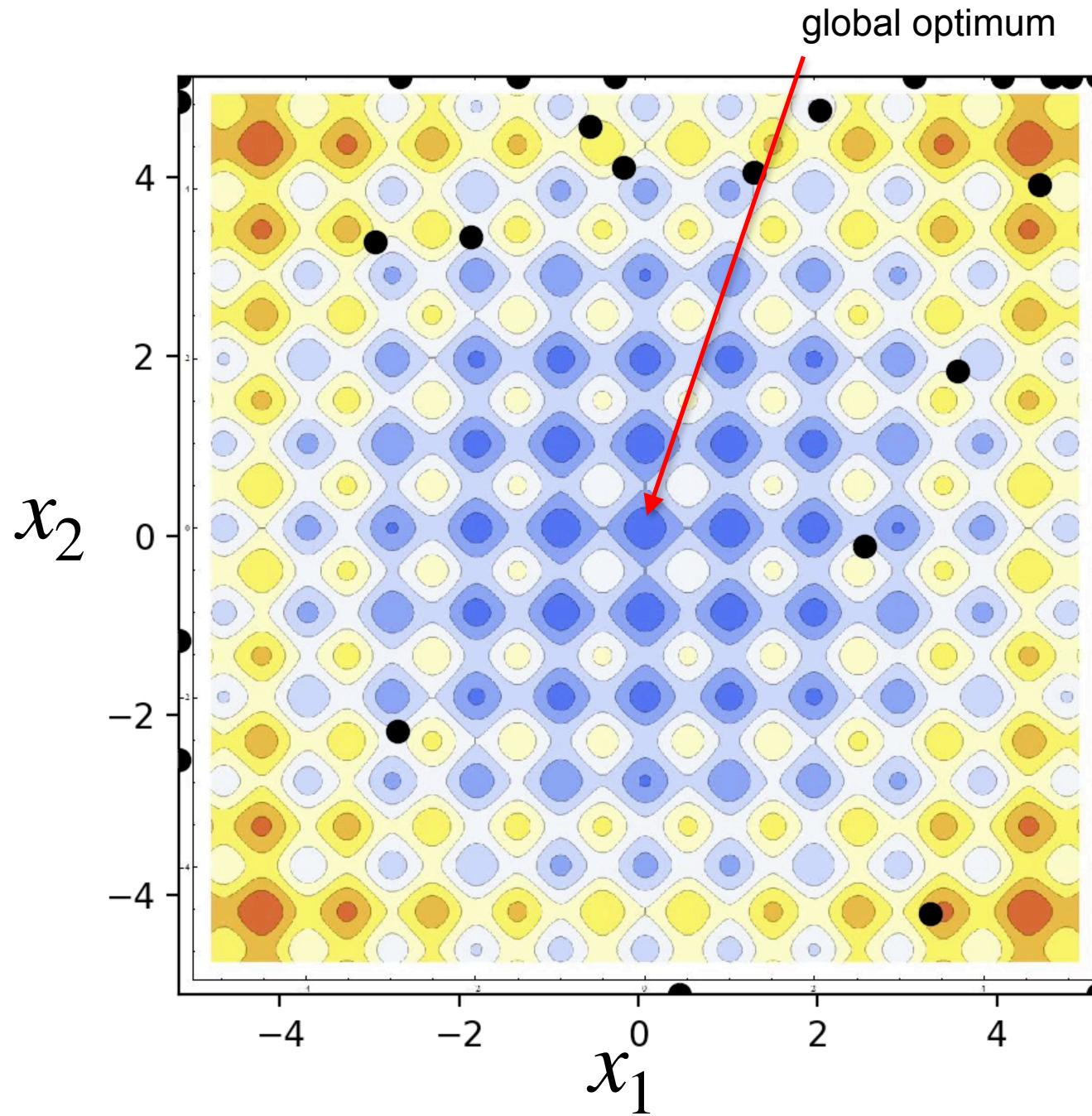


TEST FUNCTION: RASTRIGIN

global minimum: $x_1, \dots, x_n = 0, \dots, 0$

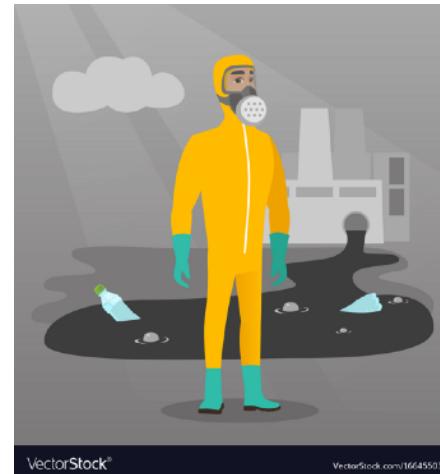
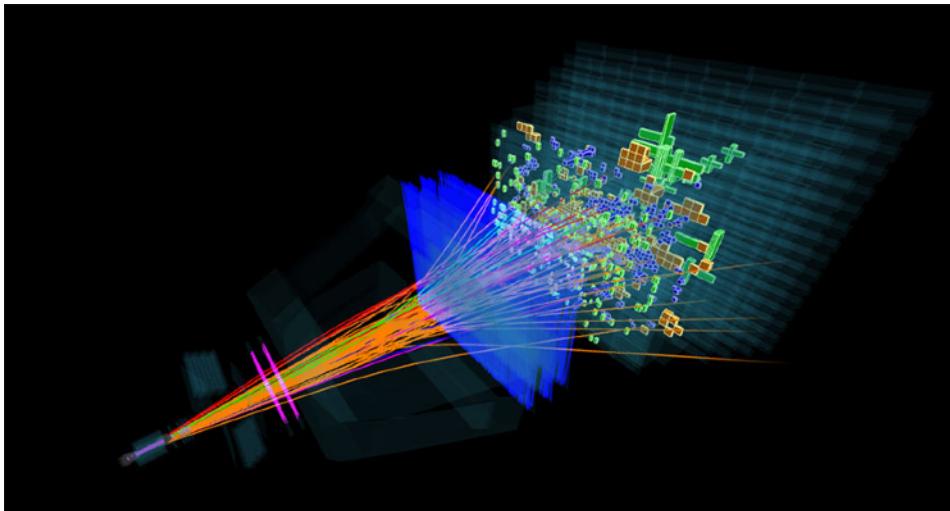
$$f(0, \dots, 0) = 0$$







EXPECTATIONS

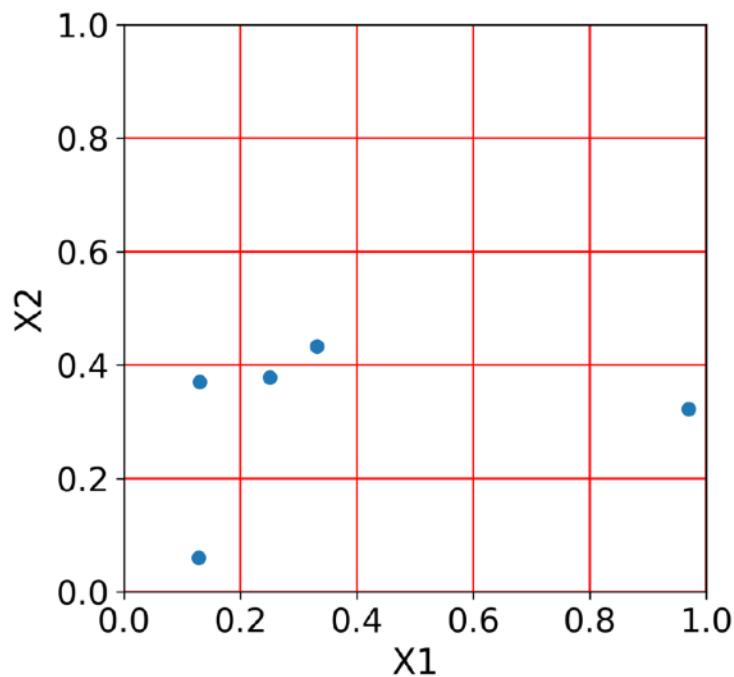


```

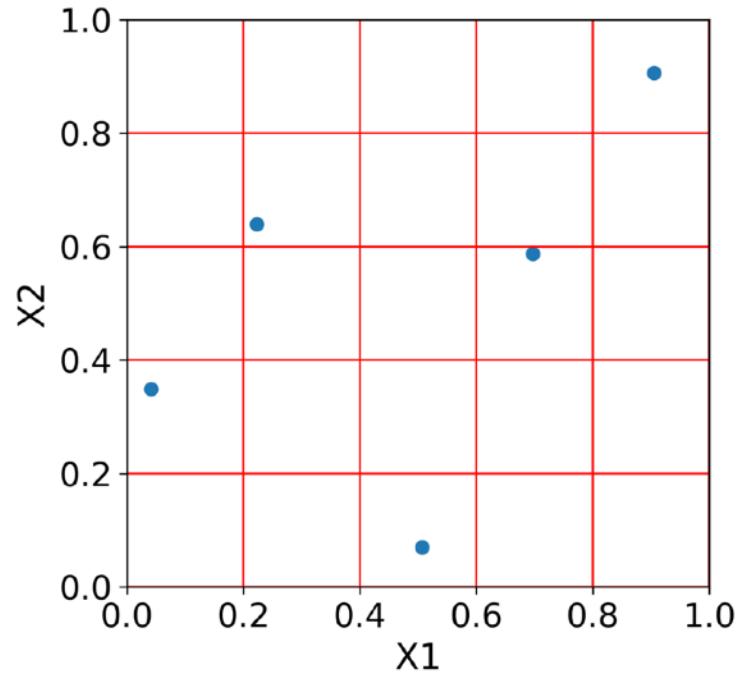
pso.py                                pso_animate.py          functions.py
1<<< "#"
2 class PSO:
3
4     def __init__(self, func_name, lb, ub, Nobj = 1, nRep = 100, bait = np.array([
5         def lhs(self):=
6
7         def _obj_wrapper(self, func, args, kwargs, x):
8             return func(x, *args, **kwargs)
9
10        def _is_feasible_wrapper(self, func, x):
11            return np.all(self.func(x)>=0)
12
13        def _cons_none_wrapper(self, x):
14            return np.array([0])
15
16        def _cons_ieqcons_wrapper(self, x):
17            return np.array([y(x, *self.args, **self.kwargs) for y in self.ieqcons()])
18
19        def _cons_f_ieqcons_wrapper(self, x):
20            return np.array(self.f_ieqcons(x, *self.args, **self.kwargs))
21
22        def Mutate(self, x, pm, mu, lb, ub):=
23
24        def Mutate_many_dimensions(self, x, pm, mu, lb, ub):=
25
26        def Dominates(self, X, Y):=
27
28        def nondomSolutions(self, X, F):=
29
30        def inverse_permutation_crowding(self, p):
31            #This function computes an "inverse permutation"
32            return np.array([p.index(l) for l in range(len(p))])
33
34        def crowding_sorting(self, archiveX, archiveF):
35            nondomN = len(archiveF)
36            crowdDist = np.zeros((nondomN, ))
37            for i in range(self.Nobj):
38                indexes = np.argsort(archiveF[:,i])
39                archivetestF = archiveF[indexes, i]
40                crowdDist = crowdDist[indexes] #(*)
41
42        def ZDT1(X):
43            ##### ZDT1 #####
44            n = X.shape[0]
45            G = 1.0
46            F1 = X[0]
47            F2 = G*(1-X[0])
48            F = np.array([F1, F2])
49
50        def Viennet:
51            F1 = 0.5*(1-X[0])
52            F2 = ((1-X[0])*(1-X[1]))**0.5
53            F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
54            F = np.array([F1, F2, F3])
55
56    def Sphere(x):
57        Sum = 0.0
58        n = float(len(x))
59        for c in x:
60            Sum += c**2
61        return Sum
62
63    def Rastrigin(x, A = 10):
64        n = float(len(x))
65        Sum = 0.0
66        for c in x:
67            Sum += c**2 - A * np.cos(2 * math.pi * c)
68        return A*n + Sum
69
70    def Holder(x):
71        return 19.2085 - np.abs(np.sin(x[0]) * np.cos(x[1]) * np.exp(abs(1 - np.sqrt(
72        x[0]**2 + x[1]**2))))
73
74    def Rosenbrock(x):
75        Sum = 0.0
76        for i in range(len(x)-1):
77            Sum += 100.0*(x[i+1]-x[i]**2)**2 + (1-x[i])**2
78        return Sum
79
80    def ZDT2(X):
81        ##### ZDT2 #####
82        n = X.shape[0]
83        G = 1.0
84        F1 = X[0]
85        F2 = G*(1-X[0])
86        F = np.array([F1, F2])
87
88    def Schaffer(x):
89        Sum = 0.0
90        n = float(len(x))
91        for c in x:
92            Sum += c**2
93        return Sum + 0.5 - 0.5 * np.cos(2 * math.pi * Sum)
94
95    def Ackley(x):
96        Sum = 0.0
97        n = float(len(x))
98        for c in x:
99            Sum += c**2
100           Sum = 0.2 * np.exp(-0.2 * np.sqrt(Sum/n)) - np.exp(secondSum/n) + 20 + math.e
101
102    def Griewank(x):
103        Sum = 0.0
104        n = float(len(x))
105        for c in x:
106            Sum += c**2
107            Sum = 1/4000 * Sum
108            Sum -= np.cos(c / np.sqrt(2))
109        return Sum + 1
110
111    def Michalewicz(x):
112        Sum = 0.0
113        n = float(len(x))
114        for c in x:
115            Sum += -np.sin(c) * np.sin(c**2 / np.pi)**2
116        return Sum
117
118    def DTLZ1(X):
119        ##### DTLZ1 #####
120        n = X.shape[0]
121        G = 1.0
122        F1 = X[0]
123        F2 = G*(1-X[0])
124        F = np.array([F1, F2])
125
126    def DTLZ2(X):
127        ##### DTLZ2 #####
128        n = X.shape[0]
129        G = 1.0
130        F1 = X[0]
131        F2 = G*(1-X[0])
132        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
133        F = np.array([F1, F2, F3])
134
135    def DTLZ3(X):
136        ##### DTLZ3 #####
137        n = X.shape[0]
138        G = 1.0
139        F1 = X[0]
140        F2 = G*(1-X[0])
141        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
142        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
143        F = np.array([F1, F2, F3, F4])
144
145    def DTLZ4(X):
146        ##### DTLZ4 #####
147        n = X.shape[0]
148        G = 1.0
149        F1 = X[0]
150        F2 = G*(1-X[0])
151        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
152        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
153        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
154        F = np.array([F1, F2, F3, F4, F5])
155
156    def DTLZ5(X):
157        ##### DTLZ5 #####
158        n = X.shape[0]
159        G = 1.0
160        F1 = X[0]
161        F2 = G*(1-X[0])
162        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
163        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
164        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
165        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
166        F = np.array([F1, F2, F3, F4, F5, F6])
167
168    def DTLZ6(X):
169        ##### DTLZ6 #####
170        n = X.shape[0]
171        G = 1.0
172        F1 = X[0]
173        F2 = G*(1-X[0])
174        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
175        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
176        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
177        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
178        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
179        F = np.array([F1, F2, F3, F4, F5, F6, F7])
180
181    def DTLZ7(X):
182        ##### DTLZ7 #####
183        n = X.shape[0]
184        G = 1.0
185        F1 = X[0]
186        F2 = G*(1-X[0])
187        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
188        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
189        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
190        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
191        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
192        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
193        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8])
194
195    def DTLZ8(X):
196        ##### DTLZ8 #####
197        n = X.shape[0]
198        G = 1.0
199        F1 = X[0]
200        F2 = G*(1-X[0])
201        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
202        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
203        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
204        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
205        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
206        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
207        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
208        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9])
209
210    def DTLZ9(X):
211        ##### DTLZ9 #####
212        n = X.shape[0]
213        G = 1.0
214        F1 = X[0]
215        F2 = G*(1-X[0])
216        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
217        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
218        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
219        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
220        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
221        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
222        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
223        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
224        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10])
225
226    def DTLZ10(X):
227        ##### DTLZ10 #####
228        n = X.shape[0]
229        G = 1.0
230        F1 = X[0]
231        F2 = G*(1-X[0])
232        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
233        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
234        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
235        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
236        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
237        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
238        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
239        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
240        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
241        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11])
242
243    def DTLZ11(X):
244        ##### DTLZ11 #####
245        n = X.shape[0]
246        G = 1.0
247        F1 = X[0]
248        F2 = G*(1-X[0])
249        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
250        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
251        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
252        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
253        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
254        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
255        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
256        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
257        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
258        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
259        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12])
260
261    def DTLZ12(X):
262        ##### DTLZ12 #####
263        n = X.shape[0]
264        G = 1.0
265        F1 = X[0]
266        F2 = G*(1-X[0])
267        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
268        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
269        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
270        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
271        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
272        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
273        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
274        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
275        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
276        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
277        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
278        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13])
279
280    def DTLZ13(X):
281        ##### DTLZ13 #####
282        n = X.shape[0]
283        G = 1.0
284        F1 = X[0]
285        F2 = G*(1-X[0])
286        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
287        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
288        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
289        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
290        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
291        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
292        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
293        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
294        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
295        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
296        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
297        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
298        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14])
299
300    def DTLZ14(X):
301        ##### DTLZ14 #####
302        n = X.shape[0]
303        G = 1.0
304        F1 = X[0]
305        F2 = G*(1-X[0])
306        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
307        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
308        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
309        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
310        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
311        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
312        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
313        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
314        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
315        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
316        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
317        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
318        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
319        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15])
320
321    def DTLZ15(X):
322        ##### DTLZ15 #####
323        n = X.shape[0]
324        G = 1.0
325        F1 = X[0]
326        F2 = G*(1-X[0])
327        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
328        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
329        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
330        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
331        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
332        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
333        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
334        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
335        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
336        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
337        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
338        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
339        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
340        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
341        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16])
342
343    def DTLZ16(X):
344        ##### DTLZ16 #####
345        n = X.shape[0]
346        G = 1.0
347        F1 = X[0]
348        F2 = G*(1-X[0])
349        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
350        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
351        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
352        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
353        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
354        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
355        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
356        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
357        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
358        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
359        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
360        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
361        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
362        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
363        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
364        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17])
365
366    def DTLZ17(X):
367        ##### DTLZ17 #####
368        n = X.shape[0]
369        G = 1.0
370        F1 = X[0]
371        F2 = G*(1-X[0])
372        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
373        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
374        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
375        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
376        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
377        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
378        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
379        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
380        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
381        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
382        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
383        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
384        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
385        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
386        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
387        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
388        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18])
389
390    def DTLZ18(X):
391        ##### DTLZ18 #####
392        n = X.shape[0]
393        G = 1.0
394        F1 = X[0]
395        F2 = G*(1-X[0])
396        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
397        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
398        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
399        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
400        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
401        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
402        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
403        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
404        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
405        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
406        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
407        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
408        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
409        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
410        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
411        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
412        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
413        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19])
414
415    def DTLZ19(X):
416        ##### DTLZ19 #####
417        n = X.shape[0]
418        G = 1.0
419        F1 = X[0]
420        F2 = G*(1-X[0])
421        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
422        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
423        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
424        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
425        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
426        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
427        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
428        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
429        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
430        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
431        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
432        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
433        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
434        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
435        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
436        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
437        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
438        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
439        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20])
440
441    def DTLZ20(X):
442        ##### DTLZ20 #####
443        n = X.shape[0]
444        G = 1.0
445        F1 = X[0]
446        F2 = G*(1-X[0])
447        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
448        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
449        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
450        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
451        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
452        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
453        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
454        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
455        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
456        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
457        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
458        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
459        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
460        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
461        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
462        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
463        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
464        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
465        F21 = 1/(1+np.exp(-5*(X[19]-0.5)))
466        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21])
467
468    def DTLZ21(X):
469        ##### DTLZ21 #####
470        n = X.shape[0]
471        G = 1.0
472        F1 = X[0]
473        F2 = G*(1-X[0])
474        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
475        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
476        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
477        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
478        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
479        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
480        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
481        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
482        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
483        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
484        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
485        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
486        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
487        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
488        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
489        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
490        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
491        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
492        F21 = 1/(1+np.exp(-5*(X[19]-0.5)))
493        F22 = 1/(1+np.exp(-5*(X[20]-0.5)))
494        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22])
495
496    def DTLZ22(X):
497        ##### DTLZ22 #####
498        n = X.shape[0]
499        G = 1.0
500        F1 = X[0]
501        F2 = G*(1-X[0])
502        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
503        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
504        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
505        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
506        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
507        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
508        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
509        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
510        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
511        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
512        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
513        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
514        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
515        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
516        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
517        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
518        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
519        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
520        F21 = 1/(1+np.exp(-5*(X[19]-0.5)))
521        F22 = 1/(1+np.exp(-5*(X[20]-0.5)))
522        F23 = 1/(1+np.exp(-5*(X[21]-0.5)))
523        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23])
524
525    def DTLZ23(X):
526        ##### DTLZ23 #####
527        n = X.shape[0]
528        G = 1.0
529        F1 = X[0]
530        F2 = G*(1-X[0])
531        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
532        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
533        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
534        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
535        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
536        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
537        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
538        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
539        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
540        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
541        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
542        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
543        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
544        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
545        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
546        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
547        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
548        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
549        F21 = 1/(1+np.exp(-5*(X[19]-0.5)))
550        F22 = 1/(1+np.exp(-5*(X[20]-0.5)))
551        F23 = 1/(1+np.exp(-5*(X[21]-0.5)))
552        F24 = 1/(1+np.exp(-5*(X[22]-0.5)))
553        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24])
554
555    def DTLZ25(X):
556        ##### DTLZ25 #####
557        n = X.shape[0]
558        G = 1.0
559        F1 = X[0]
560        F2 = G*(1-X[0])
561        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
562        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
563        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
564        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
565        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
566        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
567        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
568        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
569        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
570        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
571        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
572        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
573        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
574        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
575        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
576        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
577        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
578        F20 = 1/(1+np.exp(-5*(X[18]-0.5)))
579        F21 = 1/(1+np.exp(-5*(X[19]-0.5)))
580        F22 = 1/(1+np.exp(-5*(X[20]-0.5)))
581        F23 = 1/(1+np.exp(-5*(X[21]-0.5)))
582        F24 = 1/(1+np.exp(-5*(X[22]-0.5)))
583        F25 = 1/(1+np.exp(-5*(X[23]-0.5)))
584        F = np.array([F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24, F25])
585
586    def DTLZ26(X):
587        ##### DTLZ26 #####
588        n = X.shape[0]
589        G = 1.0
590        F1 = X[0]
591        F2 = G*(1-X[0])
592        F3 = 1/(1+np.exp(-5*(X[1]-0.5)))
593        F4 = 1/(1+np.exp(-5*(X[2]-0.5)))
594        F5 = 1/(1+np.exp(-5*(X[3]-0.5)))
595        F6 = 1/(1+np.exp(-5*(X[4]-0.5)))
596        F7 = 1/(1+np.exp(-5*(X[5]-0.5)))
597        F8 = 1/(1+np.exp(-5*(X[6]-0.5)))
598        F9 = 1/(1+np.exp(-5*(X[7]-0.5)))
599        F10 = 1/(1+np.exp(-5*(X[8]-0.5)))
600        F11 = 1/(1+np.exp(-5*(X[9]-0.5)))
601        F12 = 1/(1+np.exp(-5*(X[10]-0.5)))
602        F13 = 1/(1+np.exp(-5*(X[11]-0.5)))
603        F14 = 1/(1+np.exp(-5*(X[12]-0.5)))
604        F15 = 1/(1+np.exp(-5*(X[13]-0.5)))
605        F16 = 1/(1+np.exp(-5*(X[14]-0.5)))
606        F17 = 1/(1+np.exp(-5*(X[15]-0.5)))
607        F18 = 1/(1+np.exp(-5*(X[16]-0.5)))
608        F19 = 1/(1+np.exp(-5*(X[17]-0.5)))
609        F20 = 1/(1+np.exp(-5*(X
```

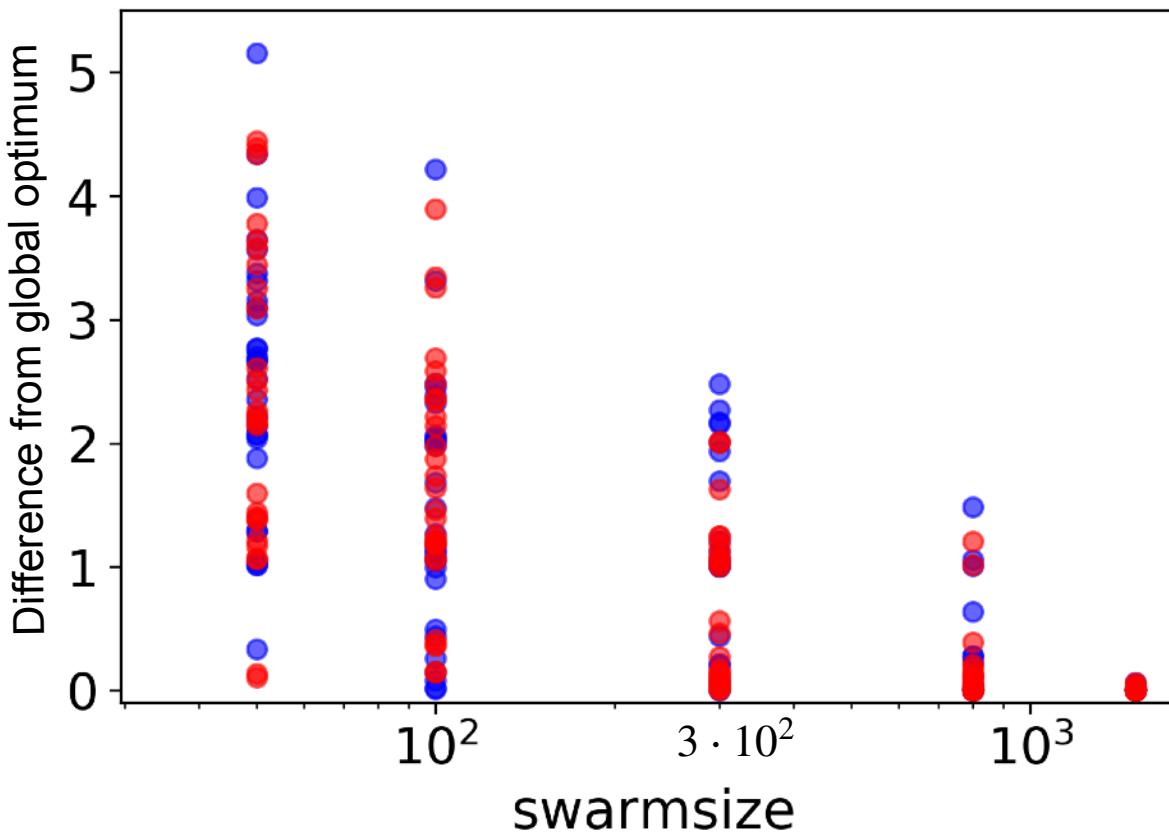
HOW TO GENERATE STARTING POSITIONS

Random Sampling



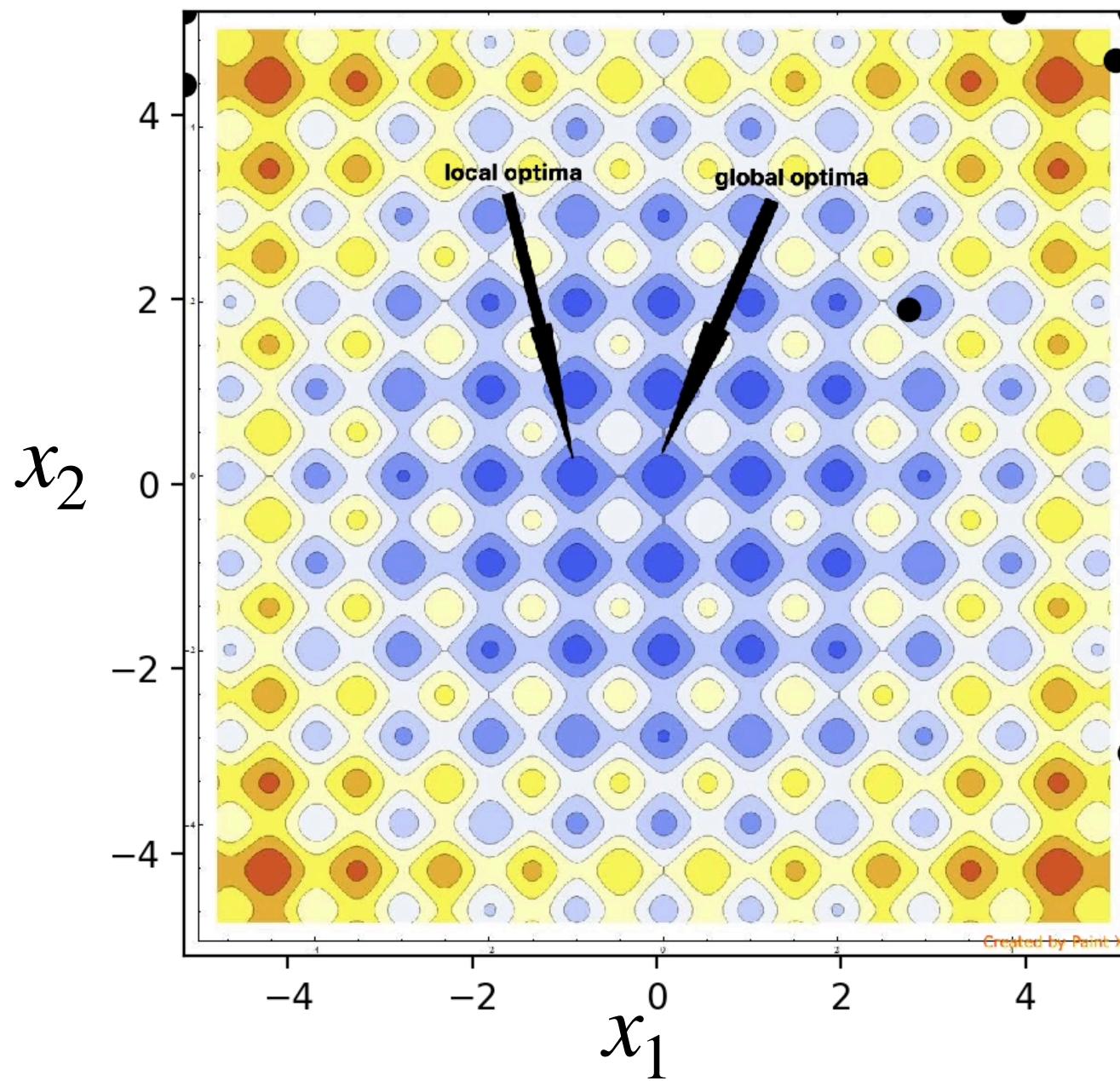
Latin Hypercube Sampling

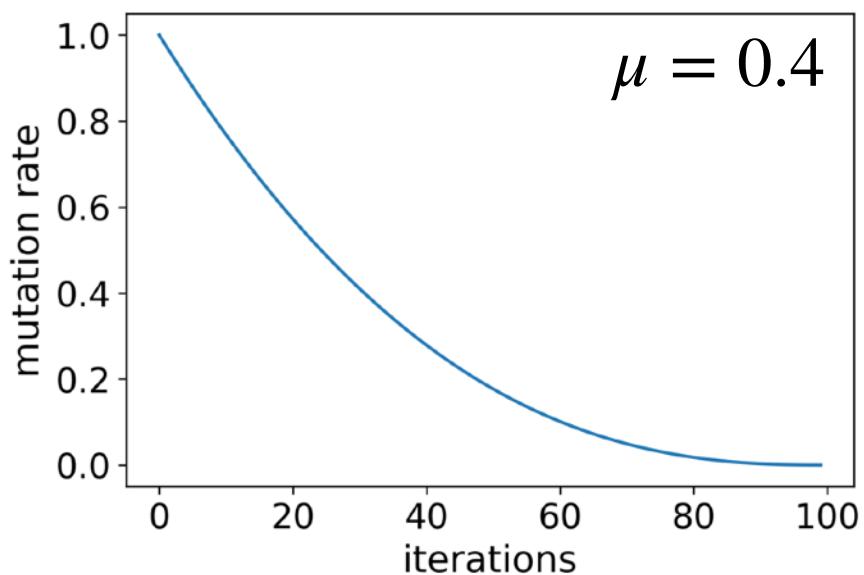
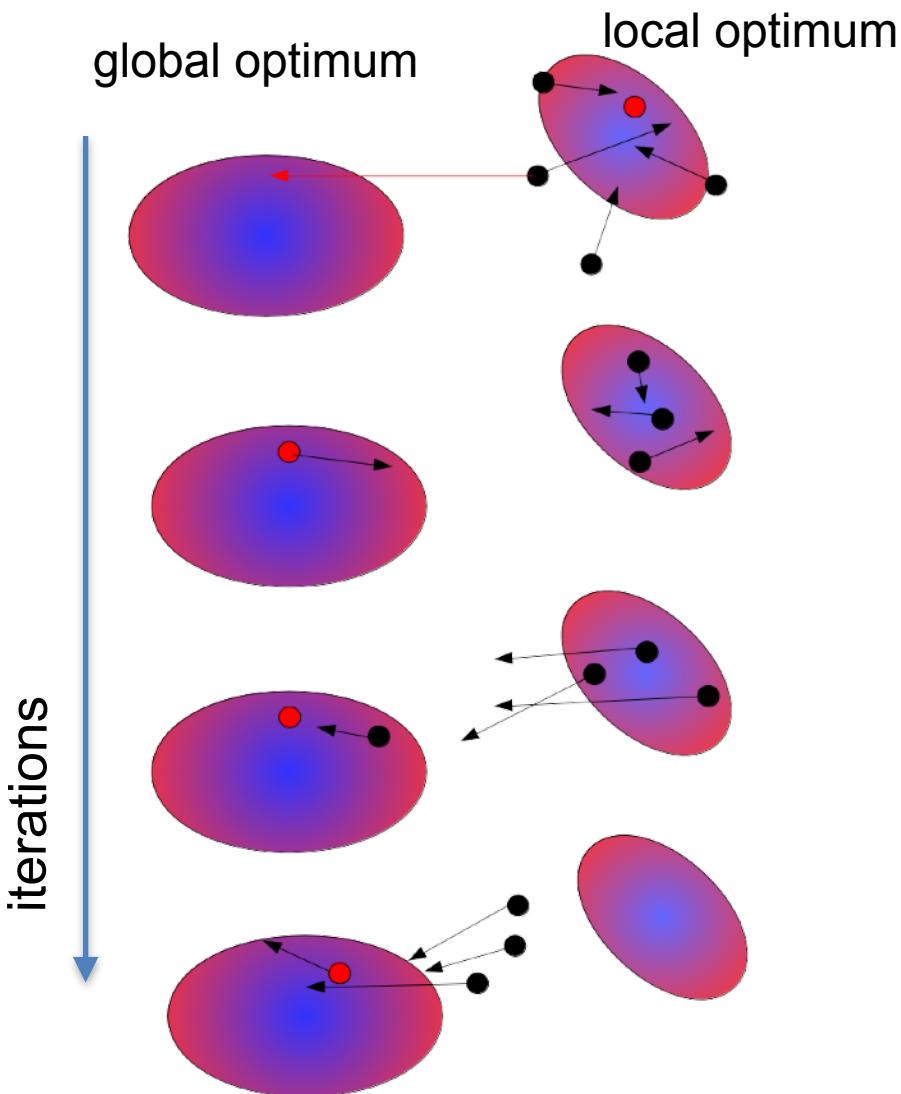




- – random
- – latin hypercube

- function: Rastrigin
- dimensions: 4
- iterations: 70





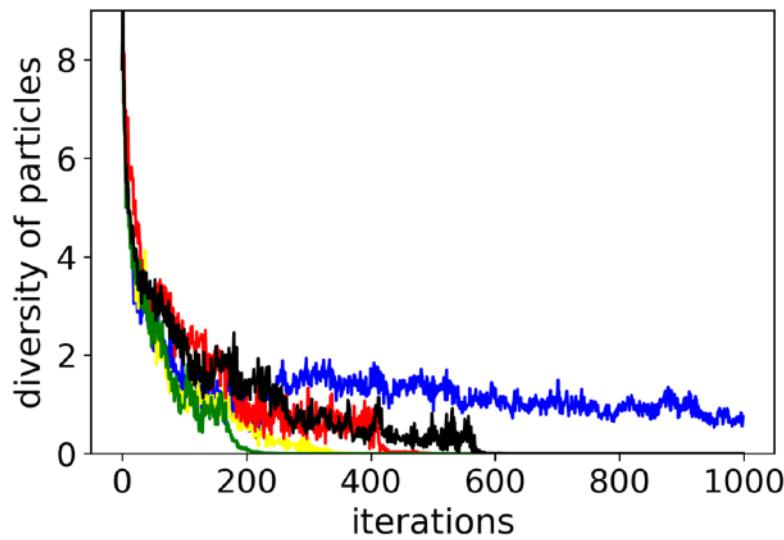
$$\text{Mut. rate} = \left(1 - \frac{n}{N}\right)^{\frac{1}{\mu}}$$

μ – mutation factor

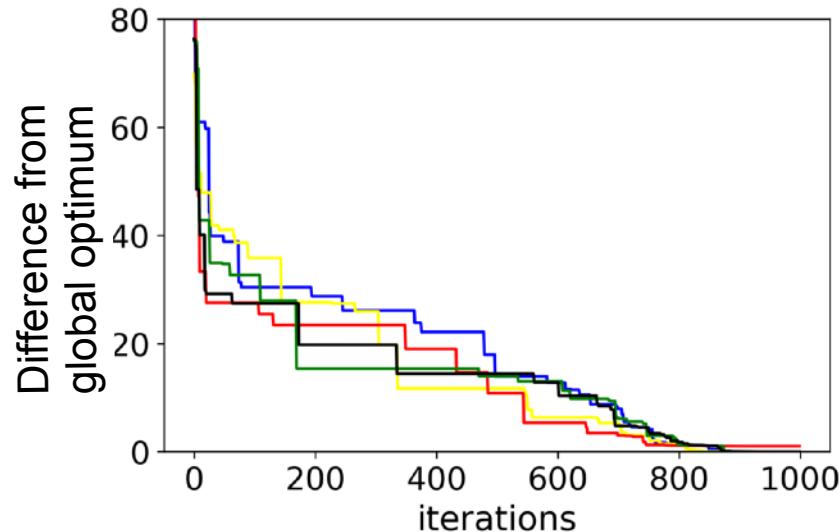
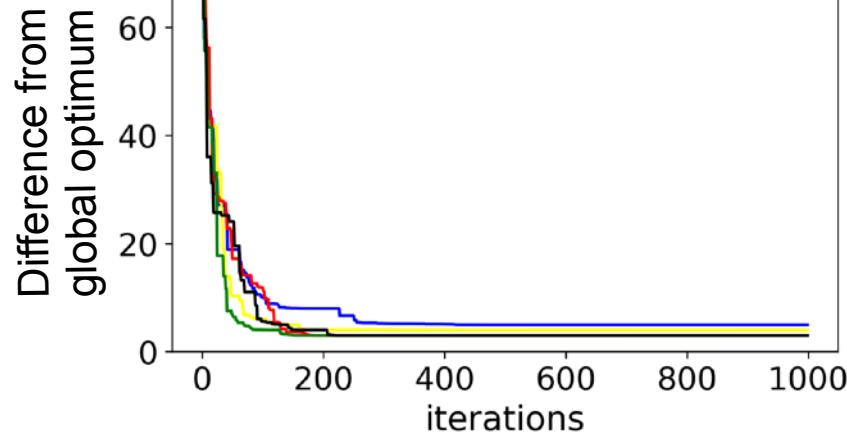
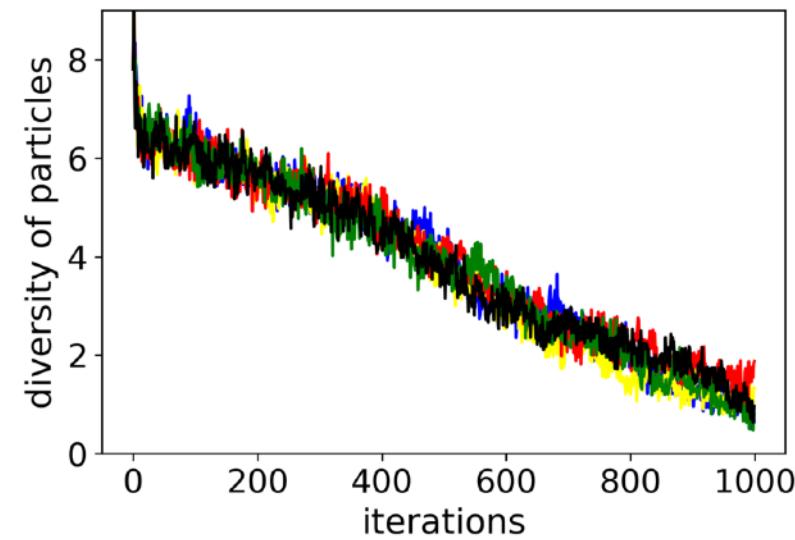
n – iteration number

N – maximum number of iterations

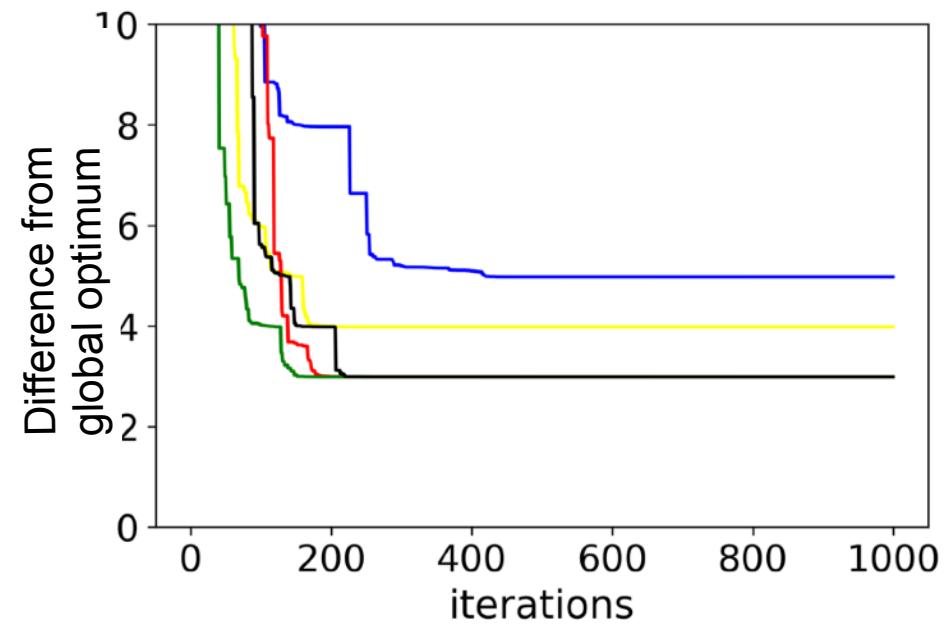
without mutation



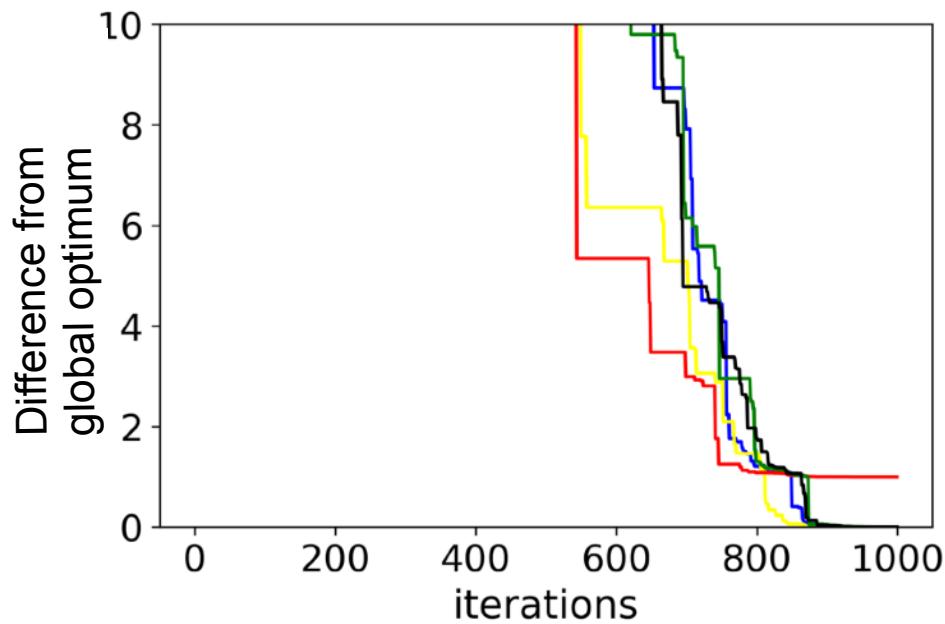
with mutation



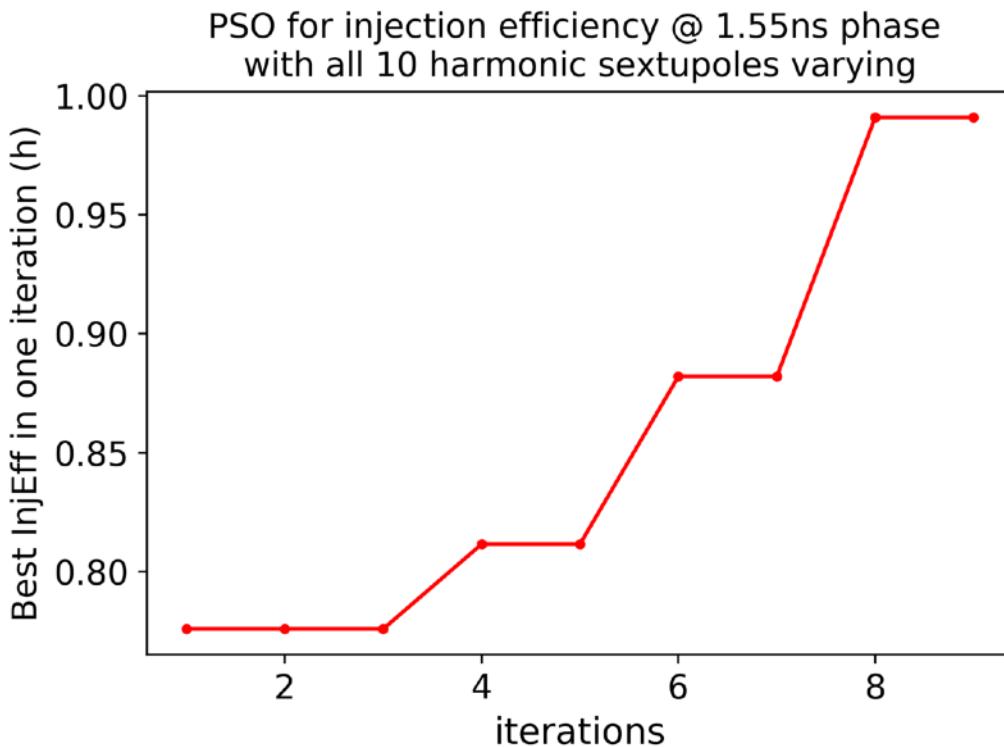
without mutation



with mutation

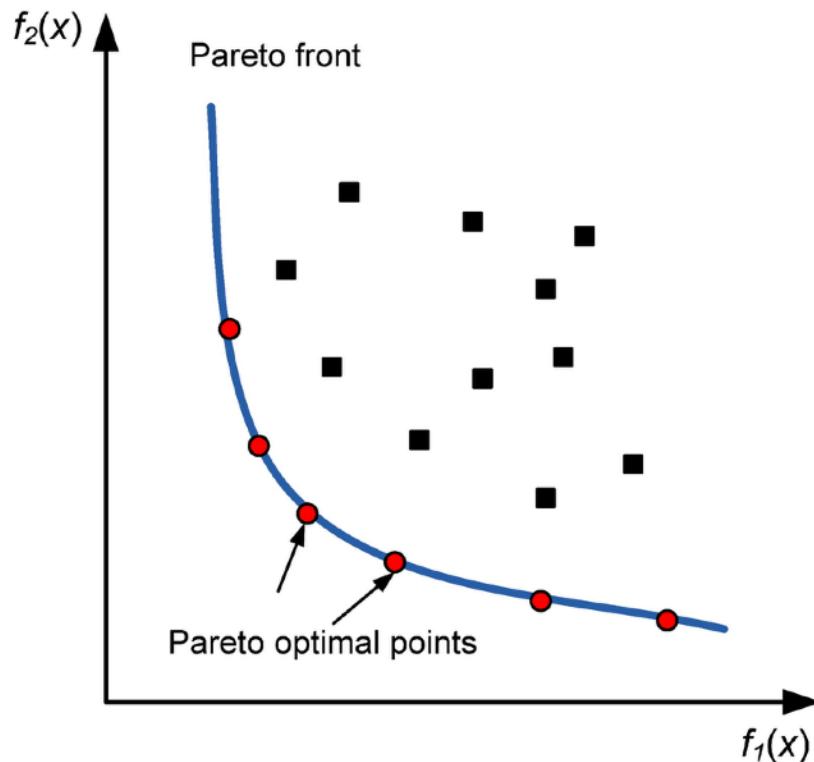


- Dimensions: 7
- Function: Rastrigin
- Swarm size: 30



Ring Sextupoles

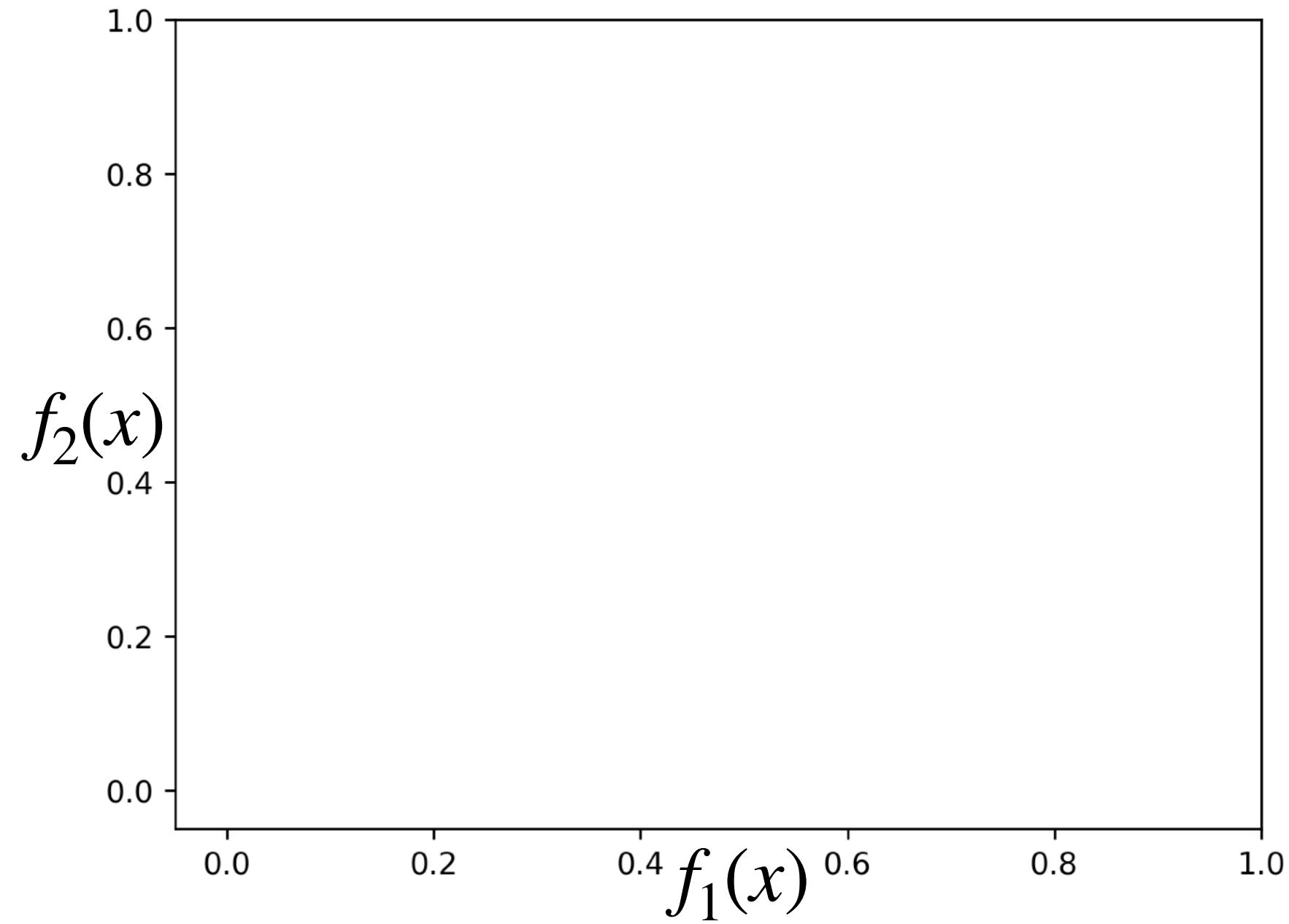
Name	S	Setpoint	Readback	Mem	C
S1PR	<input checked="" type="radio"/>	38.6272	88.5781	88.6272	<input type="checkbox"/> R
S2PDR	<input checked="" type="radio"/>	72.6533	72.6401	72.6533	<input type="checkbox"/> R
S2PTR	<input checked="" type="radio"/>	72.6533	72.6391	72.6533	<input type="checkbox"/> R
S3PDR	<input checked="" type="radio"/>	39.9495	39.9111	39.9495	<input type="checkbox"/> R
S3PTR	<input checked="" type="radio"/>	39.9495	39.9341	39.9495	<input type="checkbox"/> R
S4PDR	<input checked="" type="radio"/>	31.1680	31.1211	31.1680	<input type="checkbox"/> R
S4PTR	<input checked="" type="radio"/>	31.1680	31.1119	31.1680	<input type="checkbox"/> R
S3PD1R	<input checked="" type="radio"/>	39.9495	40.131	39.9495	<input type="checkbox"/> R
S4PD1R	<input checked="" type="radio"/>	31.1680	31.451	31.1680	<input type="checkbox"/> R
S3P1T6R	<input checked="" type="radio"/>	39.9495	40.201	39.9495	<input type="checkbox"/> R
S3P2T6R	<input checked="" type="radio"/>	39.9495	39.971	39.9495	<input type="checkbox"/> R
S4P1T6R	<input checked="" type="radio"/>	31.1680	31.301	31.1680	<input type="checkbox"/> R
S4P2T6R	<input checked="" type="radio"/>	31.1680	31.241	31.1680	<input type="checkbox"/> R



*def. **Pareto front** is the best set of solutions*

<https://github.com/txt/fss17/blob/master/review2.md>

ZDT1 function



- Implementation of LHS and mutation in PSO
- Experimental work at BESSY II
- Debugging Multi-objective PSO

$$\vec{f}_1 = (f_{11}, \dots, f_{n1})$$

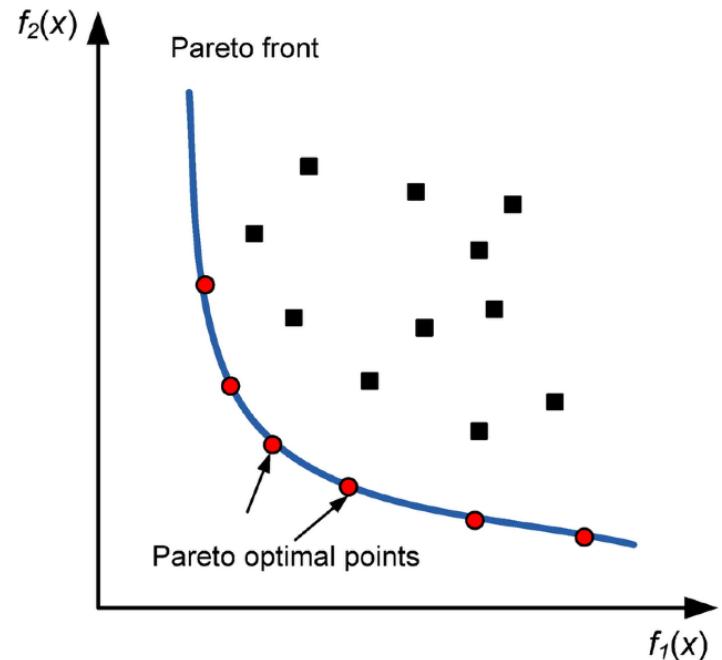
$$\vec{f}_2 = (f_{12}, \dots, f_{n2})$$

\vec{f}_1 **dominates** \vec{f}_2 if:

$$\forall i \in 1, \dots, n : f_{1i} \leq f_{2i} \text{ and } \vec{f}_1 \neq \vec{f}_2$$

def. Set is called **non-dominated set** of solutions
if no point in the set dominates other points

def. **Pareto front** is the best non-dominated set



Zitzler–Deb–Thiele's function 1