Experiment 11

Test Function

Shifted Happy Cat Function (5-D)

Expected optimum: 100 at `(88, 0.1324, -0.4233, 0.8012, -0.6352)`

```
# E11: Shifted Happy Cat Function (5-D)
# Expected optimum: 100 at `(88, 0.1324, -0.4233, 0.8012, -0.6352)`

# Dimension 5
d = 5

# Shift the parameters
x1 = x1 - 1 - 88
x2 = x2 - 1 - 0.1324
x3 = x3 - 1 + 0.4233
x4 = x4 - 1 - 0.8012
x5 = x5 - 1 + 0.6352

alpha = 1/8

xx = x1*x1 + x2*x2 + x3*x3 + x4*x4 + x5*x5

fx = ((xx-d)**2)**alpha + 1/d * (1/2 * xx + x1 + x2 + x3 + x4 + x5) + 1/2
fx = -fx
fx = fx * 10 + 100

return fx
```

Experiment Setup

- Parameter space: x1: (-2+1+88, 2+1+88), x2: (-2+1+0.1324, 2+1+0.1324), x3: (-2+1-0.4233, 2+1-0.4233), x4: (-2+1+0.8023, 2+1+0.8023), x5: (-2+1-0.6352, 2+1-0.6352)
- Number of iterations: 3 RS + 300, 3RS + 50

Results

Continuous

- 3 RS + 300

```
Best target and parameters:
{'target': 94.24479876594273, 'params': {'x1': 88.36167344675546, 'x2':
1.6311197740219607, 'x3': -0.9445289774067716, 'x4': 1.0897363341287185,
'x5': -0.8663458766741476}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 94.11239595007186, 'params': {'x1': 88.3980000422499, 'x2':
1.6300915287558222, 'x3': -0.9663013734550993, 'x4': 1.1016783928822549,
'x5': -0.8644598248140503}}
```

Discrete

Setup1: para_space = list(np.arange(-500, 501, 1000/10)) # S1

- 3 RS + 300

```
Best target and parameters:
{'target': 92.76569665591168, 'params': {'x1': 88.60000000000002, 'x2':
0.7324, 'x3': -1.4233, 'x4': 1.8023, 'x5': -0.43519999999999996}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 92.74217971837388, 'params': {'x1': 88.60000000000002, 'x2':
0.33240000000000014, 'x3': -1.4233, 'x4': 1.4023, 'x5':
0.3648000000000009}}
```

Setup3: para_space = list(np.arange(-500, 501, 1000/1000)) # S3

- 3 RS + 300

```
Best target and parameters:
{'target': 95.16948961801053, 'params': {'x1': 88.3040000000016, 'x2':
1.380400000000002, 'x3': -0.9272999999999996, 'x4': 0.966300000000001,
'x5': -0.8551999999999993}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 93.0992520329518, 'params': {'x1': 88.67600000000205, 'x2':
1.7364000000000024, 'x3': -1.1152999999999997, 'x4': 1.0663000000000011,
'x5': -0.6911999999999991}}
```

Setup5: para_space = list(np.arange(-500, 501, 1000/100000)) # S5

- 3 RS + 300

```
Best target and parameters:
{'target': 96.98887672071413, 'params': {'x1': 87.23955999999093, 'x2':
0.3907200000012584, 'x3': 0.21830000000164174, 'x4': 1.6427800000005635,
'x5': -0.7294799999990942}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 94.06474488772673, 'params': {'x1': 88.39791999994708, 'x2':
1.629880000002498, 'x3': -0.966299999999543, 'x4': 1.1015400000003979,
'x5': -0.8641999999992289}}
```

AGP

Setup1

```
def set_sparseness(i, num_iter):
    interval = (num_iter - 3) / 6

    if (i < 3 + interval):
        return 10
    elif (i < 3 + 2*interval):
        return 100
    elif (i < 3 + 3*interval):
        return 1000
    elif (i < 3 + 4*interval):
        return 10000
    elif (i < 3 + 5*interval):
        return 100000
    else:
        return None
```

- 3 RS + 300

```
Best target and parameters:
{'target': 96.30516827522642, 'params': {'x1': 88.35201420636325, 'x2':
0.6771052248278523, 'x3': -1.271442721068931, 'x4': 1.1219988759481223,
'x5': -0.3416048577016848}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 94.11534010803045, 'params': {'x1': 88.19639999999721, 'x2':
1.011999999999793, 'x3': -1.4233, 'x4': 1.373900000000045, 'x5':
-0.01520000000178404}}
```

Setup2

```
def set_sparseness(i, num_iter):
    interval = (num_iter - 3) / 4

    if (i < 3):
        return None
    elif (i < 3 + interval):
        return 10
    elif (i < 3 + 2*interval):
        return 1000
    elif (i < 3 + 3*interval):
        return 100000
    else:
        return None
```

- 3 RS + 300

```
Best target and parameters:
{'target': 96.07716288494039, 'params': {'x1': 87.78745582272133, 'x2':
0.7213738047198345, 'x3': -0.8725320136817867, 'x4': 0.7004372278152389,
'x5': 0.13155364793528546}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 93.60264821039273, 'params': {'x1': 88.02895999996105, 'x2':
0.4560400000013237, 'x3': -1.2656999999998424, 'x4': 1.5026600000005206,
'x5': 0.09476000000173013}}
```

Setup3

```
def set_sparseness(i, num_iter):
    interval = (num_iter - 3) / 6

    if (i < 3 + interval):
        return 10
    elif (i < 3 + 2*interval):
```

```
        return 20
    elif (i < 3 + 3*interval):
        return 40
    elif (i < 3 + 4*interval):
        return 80
    elif (i < 3 + 5*interval):
        return 160
    else:
        return None
```

- 3 RS + 300

```
Best target and parameters:
{'target': 96.3783212066252, 'params': {'x1': 88.25000000000028, 'x2':
0.5074000000000013, 'x3': -1.1983000000000008, 'x4': 1.2272999999999996,
'x5': -0.3852000000000044}}
```

- 3 RS + 50

```
Best target and parameters:
{'target': 95.70065581317404, 'params': {'x1': 88.20000000000027, 'x2':
0.9574000000000015, 'x3': -1.4233, 'x4': 1.4772999999999996, 'x5':
-0.1102000000000054}}
```

Thoughts

AGP's advantage: it can explore more in the first few iterations, which means it can have a better prior when doing exploitations afterwards. Therefore, I think AGP will be better for a unimodal function with an extremely large parameter space. AGP is not better than continuous/discrete BO in the case of a function with many local optima.

Try **Xin-She Yang N. 3 Function:** http://benchmarkfcns.xyz/benchmarkfcns/xinsheyangn3fcn.html