Northeastern University

专业班级 人工智能1903班 学号 20195235 姓名 王 萱

日期 2022年5月31日 成绩 指导教师 王洪峰

《求解 GA 欺骗问题》

一、实验目的

将遗传算法用于解决各种实际问题时，在求解搜索过程中可能存在某些低阶模式难以重组为期望的高阶模式，或者，低阶模式的重组引导搜索方向偏离最优解模式方向，不满足积木块假设，使得遗传算法发现全局最优解的概率大幅度减小，同时搜索效率也大大降低，这些问题统称为 GA 欺骗问题 (GA deceptive problem)。

二、实验内容

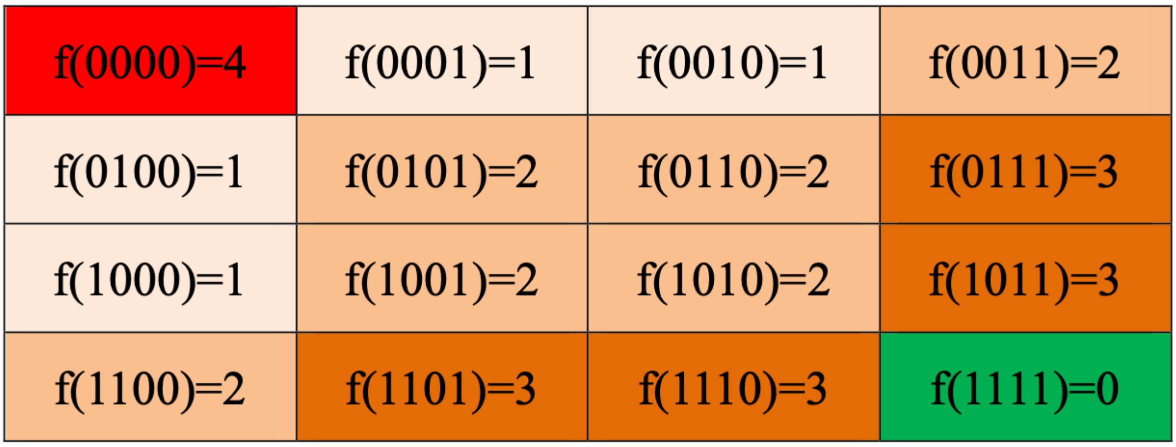
1. 实验环境

macOS 12.3

PyCharm

Visual Studio Code

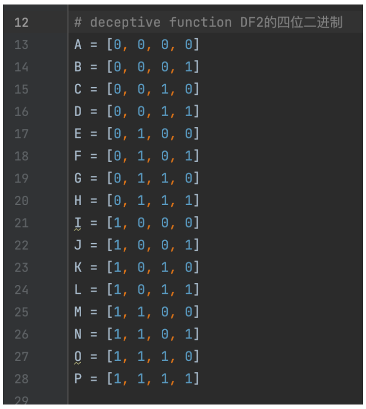
2. 实验要求题目



f(0000)=4	f(0001)=1	f(0010)=1	f(0011)=2
f(0100)=1	f(0101)=2	f(0110)=2	f(0111)=3
f(1000)=1	f(1001)=2	f(1010)=2	f(1011)=3
f(1100)=2	f(1101)=3	f(1110)=3	f(1111)=0

图 1 题目要求，最大值为 100

3. 将函数与值格式化



```
# deceptive function DF2的四位二进制
A = [0, 0, 0, 0]
B = [0, 0, 0, 1]
C = [0, 0, 1, 0]
D = [0, 0, 1, 1]
E = [0, 1, 0, 0]
F = [0, 1, 0, 1]
G = [0, 1, 1, 0]
H = [0, 1, 1, 1]
I = [1, 0, 0, 0]
J = [1, 0, 0, 1]
K = [1, 0, 1, 0]
L = [1, 0, 1, 1]
M = [1, 1, 0, 0]
N = [1, 1, 0, 1]
O = [1, 1, 1, 0]
P = [1, 1, 1, 1]
```

图 2 格式化数据

4. 遗传算法选择操作

```python
# GA选择操作
def Choose(Group, fitness):  # 遗传操作-轮盘赌选择法

    Target = np.random.choice(np.arange(POP_NUM), size=POP_NUM, replace=True,
                              p=fitness / fitness.sum())  # 计算选择概率
    return Group[Target]
```

图 3 选择操作

5. 交叉操作

```python
# GA交叉操作
def Crossover(LastChrom, Entity):
    if np.random.rand() < CROSS_RATE:

        m = np.random.randint(0, POP_NUM, size=1)
        Cross_Cut1 = int(np.random.randint(0, Chromosome_size, size=1))  # 选择交叉切点
        Cross_Cut2 = int(np.random.randint(0, Chromosome_size, size=1))
        if (Cross_Cut1 < Cross_Cut2):
            LastChrom[Cross_Cut2 - Cross_Cut1:Chromosome_size - Cross_Cut1] = Entity[m, Cross_Cut2:]
        else:
            LastChrom[Cross_Cut1 - Cross_Cut2:Chromosome_size - Cross_Cut2] = Entity[m, Cross_Cut1:]

    return LastChrom
```

图 4 交叉操作

6. 变异操作

```python
# GA变异操作
def variation(NewChrom):
    for point in range(Chromosome_size):
        if np.random.rand() < MUT_RATE:
            NewChrom[point] = 1 if NewChrom[point] == 0 else 0
    return NewChrom  # 返回新子代
```

图 5 变异操作

7. 迭代操作

```python
for n in range(ITER_NUM):  # 迭代计算
    F_values = binToDec(Group)  # 计算适应值对应的值

    Group = Choose(Group, np.array(F_values))
    pop_copy = Group.copy()

    for LastChrom in Group:
        NewChrom = Crossover(LastChrom, Group)
        NewChrom = variation(NewChrom)

        Gene = NewChrom_value(NewChrom)
        if (Gene > np.median(F_values)):  # 子代适应值超过中位数则接受
            pop_copy[np.argmin(F_values), :] = NewChrom  # 将原种群中的适应值最小的一个替换掉
            F_values[np.argmin(F_values)] = Gene  # 更新适应值

    Group = pop_copy.copy()

    Fit_value.clear()  # 适应值清零
    values_new = binToDec(pop_copy)
    x = np.argmax(values_new)
    print("Best Chromosome: ""\n", Group[x])  # 打印最大值对应的基因
```

图 6 迭代操作

8. 初始迭代状态

```
/usr/local/bin/python3.9 /Users/jack/Desktop/repo/IntelligentOptimizationMethods/e
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0]
最大函数值:  64
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0]
最大函数值:  64
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0]
最大函数值:  67
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 0]
最大函数值:  64
最佳基因组:
 [1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0
 0 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 1
 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 0 1 1]
最大函数值:  65
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1]
最大函数值:  66
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 1 0
 0 0 0 0 0 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0
```

图 7 刚开始迭代搜索

9. 迭代得到最大值 100

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]
最大函数值:  99
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]
最大函数值:  99
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]
最大函数值:  99
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]
最大函数值:  99
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0]
最大函数值:  99
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
最大函数值:  100
最佳基因组:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
最大函数值:  100
最佳基因组:
```

图 8 迭代得到最大值 100

三、实验体会

对于大多数具有 GA 欺骗性的问题, GA 虽然能够获得全局最优解, 但发现全局最优解的概率会大幅度减小, 同时搜索过程的效率也会大大降低, 直接影响着 GA 的运行性能。这其中许多严重的 GA 欺骗性问题被归结为 GA 难题, 使 GA 的应用受到一定的局限。