作业

Homework2

庄镇华 502022370071

A HSEA Homework Assignment

2022 年 11 月 13 日

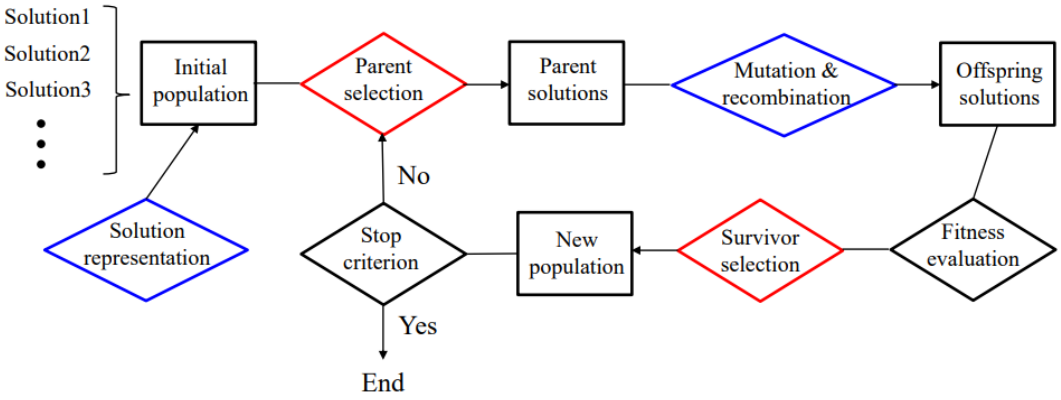

图 1: 演化算法流程图

1 求解子模优化问题 MaxCut

问题 1

实现演化算法求解子模优化问题 MaxCut

- 你需要实现演化算法：定义解与种群以及设计演化算子
- 你需要汇报相应的实验结果：演化算法参数设置（种群大小，算子参数，迭代轮数）；问题参数设置（规模等）；画出以评估次数（或迭代轮数）为横轴，fitness 为纵轴的曲线，以观察算法的效果

解答：

1.1 问题定义

最大割问题。给定一个无向加权图 $G = (V, E, w)$，权重为 $w : E \to \mathbb{R}_{\geq 0}$，目标是选择 $V_1 \subseteq V$ 使得 V_1 和 $V_2 = V \backslash V_1$ 之间边的权重之和最大。

对于给定的解 $x \in \{0\ 1\}^n$，其中 $n = |V|$，我们有 $V_1(x) = \{v_i | x_i = 1\}$ 和 $V_2(x) = \{v_i | x_i = 0\}$。设 $C(x) = \{e \in E | e \cup V_1(x) \neq \emptyset \wedge e \cup V_2(x) \neq \emptyset\}$ 是给定解 x 对应的割，我们的目标是最大化

$$f'(x) = \sum_{e \in C(x)} w(e)$$

1.2 定义解与种群

常见的演化算法流程如图1所示，由于最大割问题属于离散优化问题，且解空间 $\{0, 1\}^n$ 中的每个解都是可行的，因此本次实验采取二进制表示方法，本次实验采用的算法和经典的遗传算法类似，但一些细节略有不同。

设 n 为给定图的结点数，则解 $x \in \{0, 1\}^n$，将归属于 1 的结点和归属于 0 的结点之间的边定义为割边，则所有割边的和就是需要优化的目标，也是适应度函数的实现方法。

个体即解采用二进制表示，解 x 的每个维度都有 50% 的可能等于 0 或 1。

```
class Individual:
  def __init__(self, genotype = None):
    if genotype is None:
      genotype = []
    self.genotype = np.array(genotype)
    self.fitness = 0

  def initialize(length):
    individual = Individual()
    individual.genotype = np.random.choice((0, 1), p = (0.5, 0.5), size = length)
    return individual
```

种群的初始化规模设置为 100，每个个体的每个维度都进行随机初始化。

```
self.population_size = 100
def initialize_population(self): # 初始化种群
  self.population = [Individual.initialize(self.n_nodes) for _ in range(self.
    population_size)]
  for individual in self.population:
    self.get_fitness(individual)
```

1.3 设计演化算子

演化算子包含：亲代选择算子、变异算子、交叉算子、生存选择算子。此外，停止准则为达到 5000 迭代次数后停止算法。

亲代选择算子。亲代选择算子选取均匀选择算子，即从当前种群中随机选取个体，

$$P_{US}(i) = \frac{1}{\mu}$$

在执行 λ 次均匀选择后，交配池中第 j 个个体被选择的期望次数是 $\frac{\lambda}{\mu}$。

```
def parents_selection(self, choice = 'uniform'):
  if choice == 'uniform':
    parents_pool = []
    N = len(self.population)
    for _ in range(N):
      parents_pool.append(self.population[np.random.randint(N)])
    return parents_pool
```

变异算子。变异算子选取 bit-wise 变异算子，即个体的每一位都以概率 p_m 进行翻转，翻转位数是一个随机变量，其概率公式为

$$P(X = k) = C_n^k p_m^k (1 - p_m)^{n-k}$$

即 X 满足二项分布 $B(n, p_m)$。翻转次数的期望为 $E[X] = np_m$，当 $p = \frac{1}{n}$ 时值为 1。

```
def bitwise_mutation(ind_list):
  l = len(ind_list[0].genotype)
  p = 1 / l
  for ind in ind_list:
    m = np.random.choice((0,1), p=(1-p, p), size=l)
```

```
    flip = 1 - ind.genotype
    ind.genotype = np.where(m, flip, ind.genotype)
  return ind_list
```

交叉算子。交叉算子选取均匀交叉算子，即对于第一个子代，每个基因都有 p 的概率遗传自第一个亲代个体，剩余的基因遗传自第二个亲代个体；对于第二个子代，其基因型为第一个子代的翻转。

```
def uniform_crossover(individual_a: Individual, individual_b: Individual, p = 0.5)
    :
  l = len(individual_a.genotype)
  offspring_a = Individual(l)
  offspring_b = Individual(l)

  m = np.random.choice((0,1), p=(p, 1-p), size=l)
  offspring_a.genotype = np.where(m, individual_a.genotype, individual_b.genotype)
  offspring_b.genotype = np.where(1 - m, individual_a.genotype, individual_b.
    genotype)
  return [offspring_a, offspring_b]
```

生存选择算子。生存选择算子选取循环锦标赛算子，即每个个体 x 都和从当前种群和后代中随机选择的 q 个其他个体进行适应度评估比赛；对于每次比赛，如果 x 的适应度高于其对手，就会赢得比赛；最终获胜次数最高的 100 个个体会被保留下来形成下一代种群。

```
def tournament_selection(population, offspring):
  selection_pool = np.concatenate((population, offspring), axis = None)
  tournament_size = 4
  assert len(selection_pool) % tournament_size == 0, "Population size should be a
    multiple of 2"

  selection = []
  number_of_rounds = tournament_size // 2
  for _ in range(number_of_rounds):
    number_of_tournaments = len(selection_pool) // tournament_size
    order = np.random.permutation(len(selection_pool))
    for j in range(number_of_tournaments):
      indices = order[tournament_size * j : tournament_size * (j + 1)]
      best = select_best_solution(selection_pool[indices])
      selection.append(best)
  assert(len(selection) == len(population))
  return selection
```

1.4 实验结果

问题参数设置（图规模）如表1所示，相应的实验结果如表2所示，基础演化算法迭代次数和适应度的曲线图如图2所示，可以看到在大多数图中算法仅仅迭代了不到 1000 步就陷入局部最优解。

基础演化算法参数设置如下：种群大小为 100，迭代次数为 5000 轮，亲代选择使用均匀选择算子，变异使用 bitwise 变异算子，交叉采用均匀交叉算子，其交叉概率 $p = 0.5$，生存选择采用循环锦标赛算子，保留的种群规模同样为 100。

复现方式见 readme.md 文件。

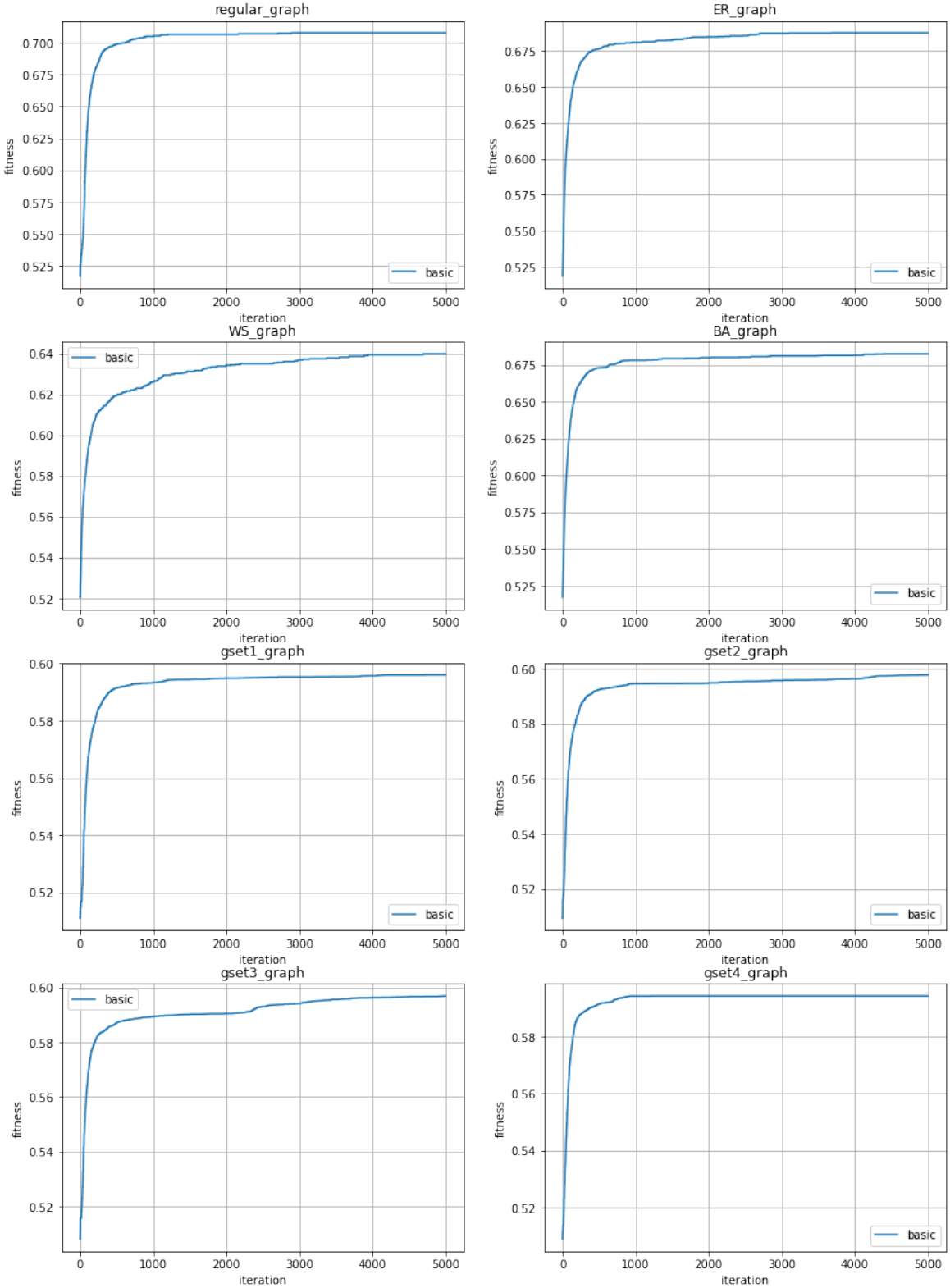

图 2: 基础演化算法效果图

	G1	G2	G3	G4	正则图	ER 随机图	WS 小世界网络图	BA 无标度网络图
结点数	800	800	800	800	1000	1000	1000	1000
边数	19176	19176	19176	19176	5000	5957	5000	5964

表 1: 不同图的参数设置

	G1	G2	G3	G4	正则图	ER 随机图	WS 小世界网络图	BA 无标度网络图
fitness	0.5960	0.5977	0.5969	0.5941	0.7076	0.6876	0.6398	0.6824

表 2: 基础演化算法在不同图上的性能

2 设计更适合 MaxCut 问题的演化算法

问题 2

演化算法改进：如何设计更适合 MaxCut 问题的演化算法？

- 可能的改进方向：演化算子改进以及参数优化
- 你需要汇报相应的实验结果：演化算法参数设置（种群大小，算子参数，迭代轮数）；问题参数设置（规模等）；如果参考了论文，请引用并标明；画出以评估次数（或迭代轮数）为横轴，fitness 为纵轴的曲线，以观察算法的效果，与任务一中的原始算法对比

解答：

2.1 算法改进方向

2.1.1 演化算子改进

亲代选择算子。亲代选择算子由均匀选择算子改进为基于轮盘赌注采样的适应度比例（FPS）选择算子，即选择第 i 个个体的概率是

$$P_{FPS}(i) = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

其中 μ 代表种群规模数目。轮盘赌注采样算法如下所示，且最终第 i 个个体出现在交配池中的期望次数为 $\lambda P_{sel}(i)$。

```
def parents_selection(self, choice = 'fitness'):
  if choice == 'fitness': # Roulette wheel sampling
    wheel = np.array([_.fitness for _ in self.population])
    wheel /= wheel.sum()
    N = len(self.population)
    parents_pool = []
    for _ in range(N):
      rand_val, p = random.random(), 0
      for i in range(N):
        p += wheel[i]
        if p >= rand_val:
```

算法 1 轮盘赌注算法

1:$cur = 1$
2: 当 $cur \leq \lambda$ 时，循环
3: 从 $[0, 1]$ 中均匀随机取样 r
4: $r = 1$
5: 当 $a_i < r$ 时，循环
6: $i = i + 1$
7: 结束循环
8: $mating\_pool[cur] = parents[i]$
9: $cur = cur + 1$
10: 结束循环

```
            parents_pool.append(self.population[i])
            break
    return parents_pool
```

变异算子。变异算子由 bit-wise 变异算子改进为 one-bit 变异算子，即翻转随机选择的位置，这种变异行为会保证每次发生后都必然有 1 位被翻转。

```
def onebit_mutation(ind_list):
  for ind in ind_list:
    pos =  np.random.randint(len(ind.genotype))
    ind.genotype[pos] = 1 - ind.genotype[pos]
  return ind_list
```

生存选择算子。生存选择算子由循环锦标赛算子改进为基于适应度的 $(\lambda + \mu)$ 选择算子，即当前种群和 λ 个子代个体中适应度最高的 μ 个个体会被保留下来形成下一代种群。这种方法可以保留生成的最好个体。

```
def best_selection(population, offspring):
    selection_pool = np.concatenate((population, offspring), axis = None).tolist()
    selection_pool.sort(key = lambda x : x.fitness, reverse = True)
    return selection_pool[ : len(population)]
```

2.1.2 参数优化改进

参数调节方面，使用网格调参方法调节均匀交叉算子的概率 $p = 0.6$，最终经过演化算子改进和参数优化改进得到了最终版本，实验取得的效果见下一小节。

2.2 实验结果对比

	G1	G2	G3	G4	正则图	ER 随机图	WS 小世界网络图	BA 无标度网络图
basic_fitness	0.5960	0.5977	0.5969	0.5941	0.7076	0.6876	0.6398	0.6824
advanced_fitness	0.5960	0.5982	0.5990	0.5992	0.7152	0.6985	0.6400	0.6856

表 3: 改进演化算法在不同图上的性能对比

相应的问题参数设置和基础版保持一致，可见表1。改进算法与基础算法的对比实验结果如图3所示，横轴代表迭代轮数，纵轴代表适应度，可以看到改进算法的效果相对于基础算法有了一定提升，说明改进起到了很好的效果。

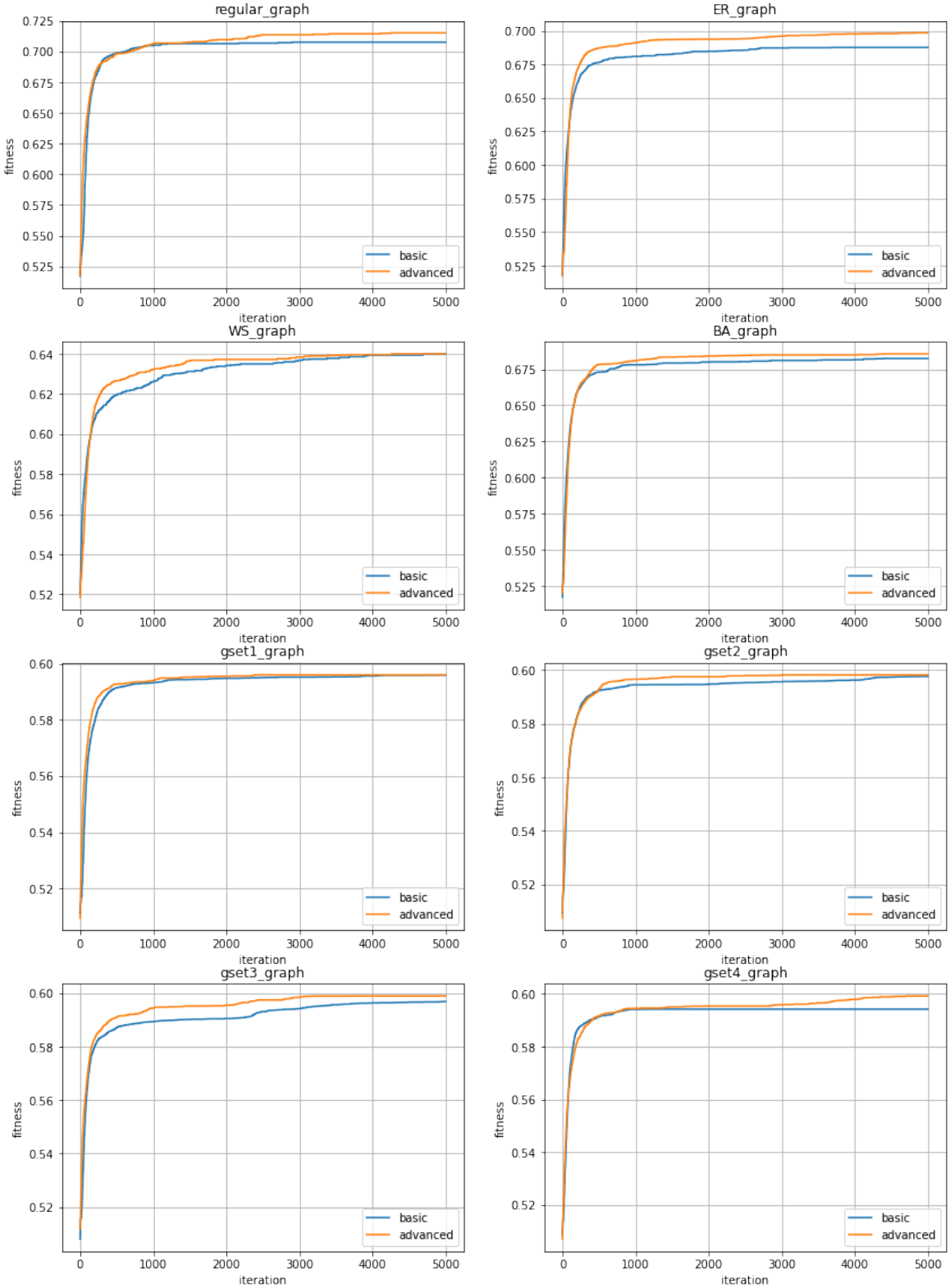

图 3: 改进演化算法效果对比图

改进演化算法参数设置如下：种群大小为 100，迭代次数为 5000 轮，亲代选择使用基于轮盘赌注采样的适应度比例选择算子，变异使用 onebit 变异算子，确保每次都有一位翻转，交叉采用均匀交叉算子，其交叉概率 $p = 0.6$，生存选择采用基于适应度的 $\lambda + \mu$ 算子，确保生成的最好个体被保留下来，保留的种群规模同样为 100。

复现方式见 readme.md 文件。