湖北大学
HUBEI UNIVERSITY

实验四　遗传算法

本周实验内容

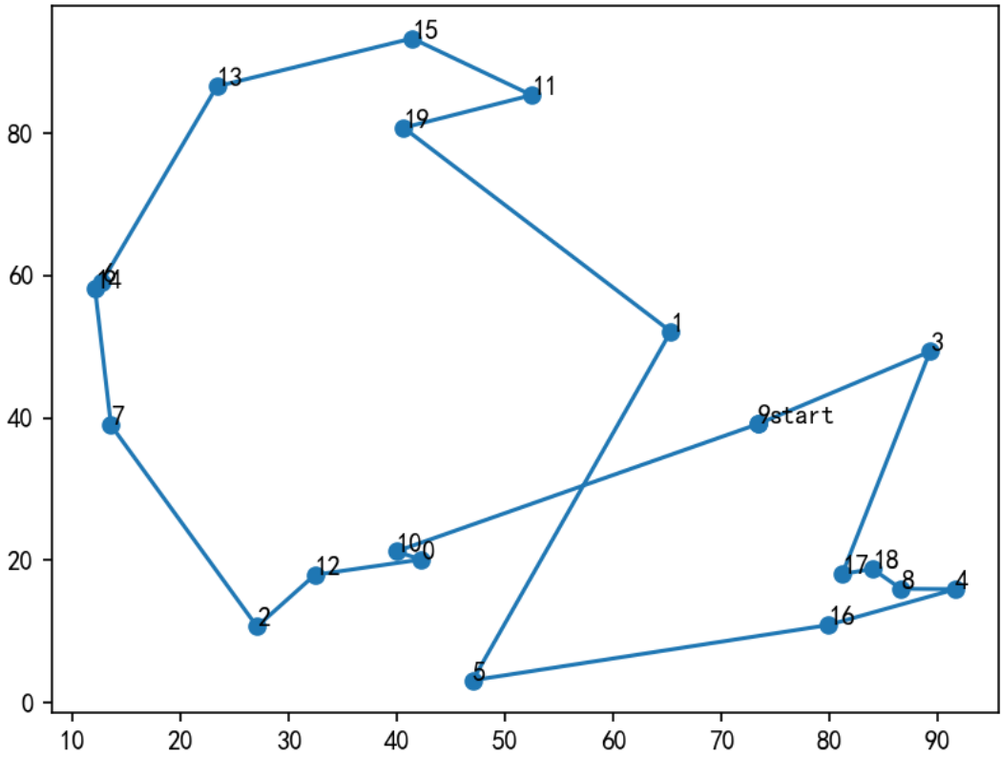
- 使用遗传算法解决旅行商问题（Traveling Salesman Problem）
- 完成并提交实验报告（4月30日上课前交给班长）

实验要求

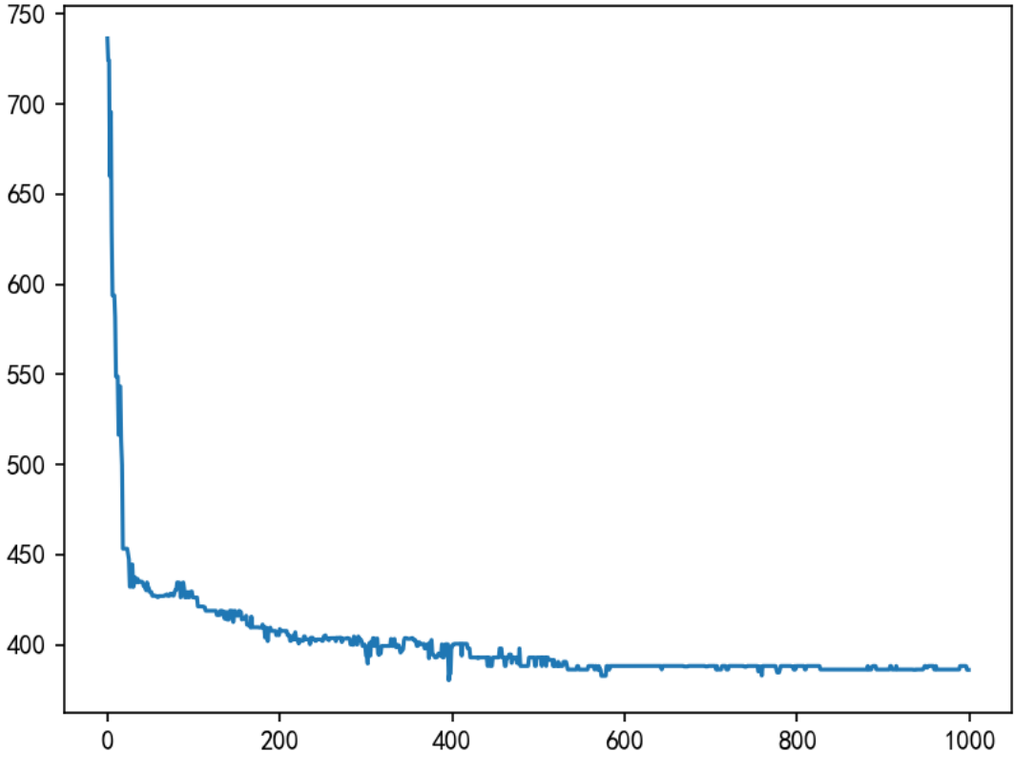
- 用遗传算法求解不同规模（如10个城市，20个城市，100个城市）的旅行商问题：
  - ✓ 城市的x坐标和y坐标在区间 $[0,100]$ 内随机生成；
  - ✓ 设置不同的种群规模、交叉概率和变异概率；
  - ✓ 使用Matplotlib画出算法求得的路线，以及路线长度随着迭代次数的变化。

实验要求

例：算法求得的路线

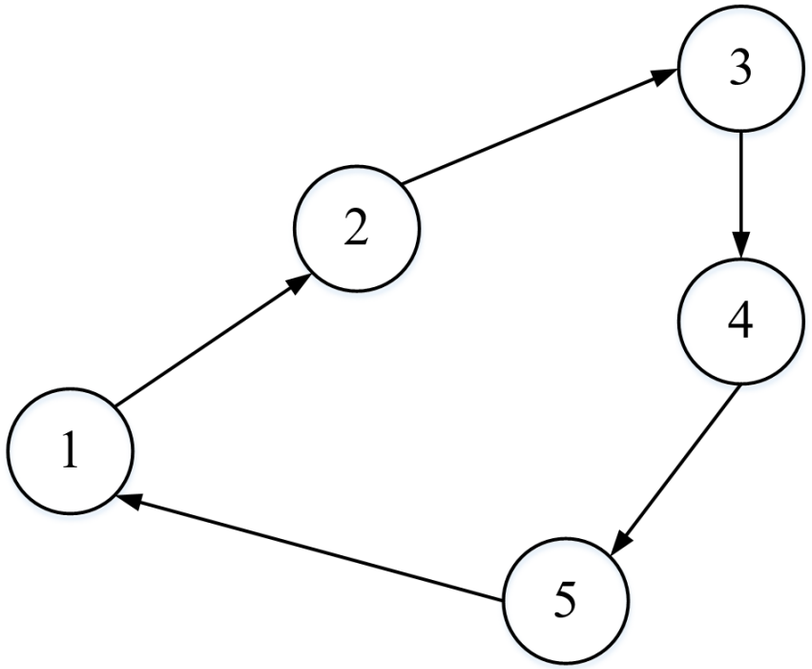


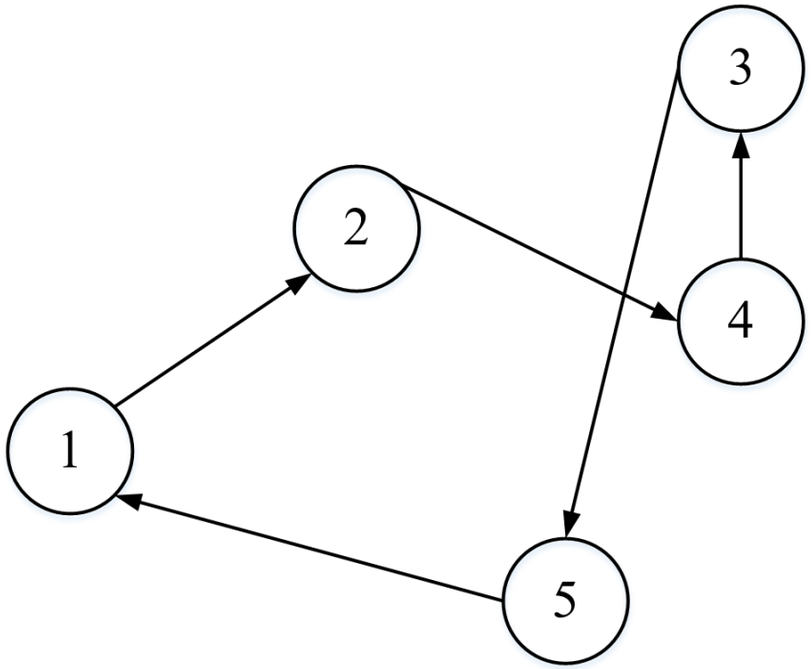
例：路线长度随着迭代次数的变化



遗传算法 (Genetic Algorithm, GA)

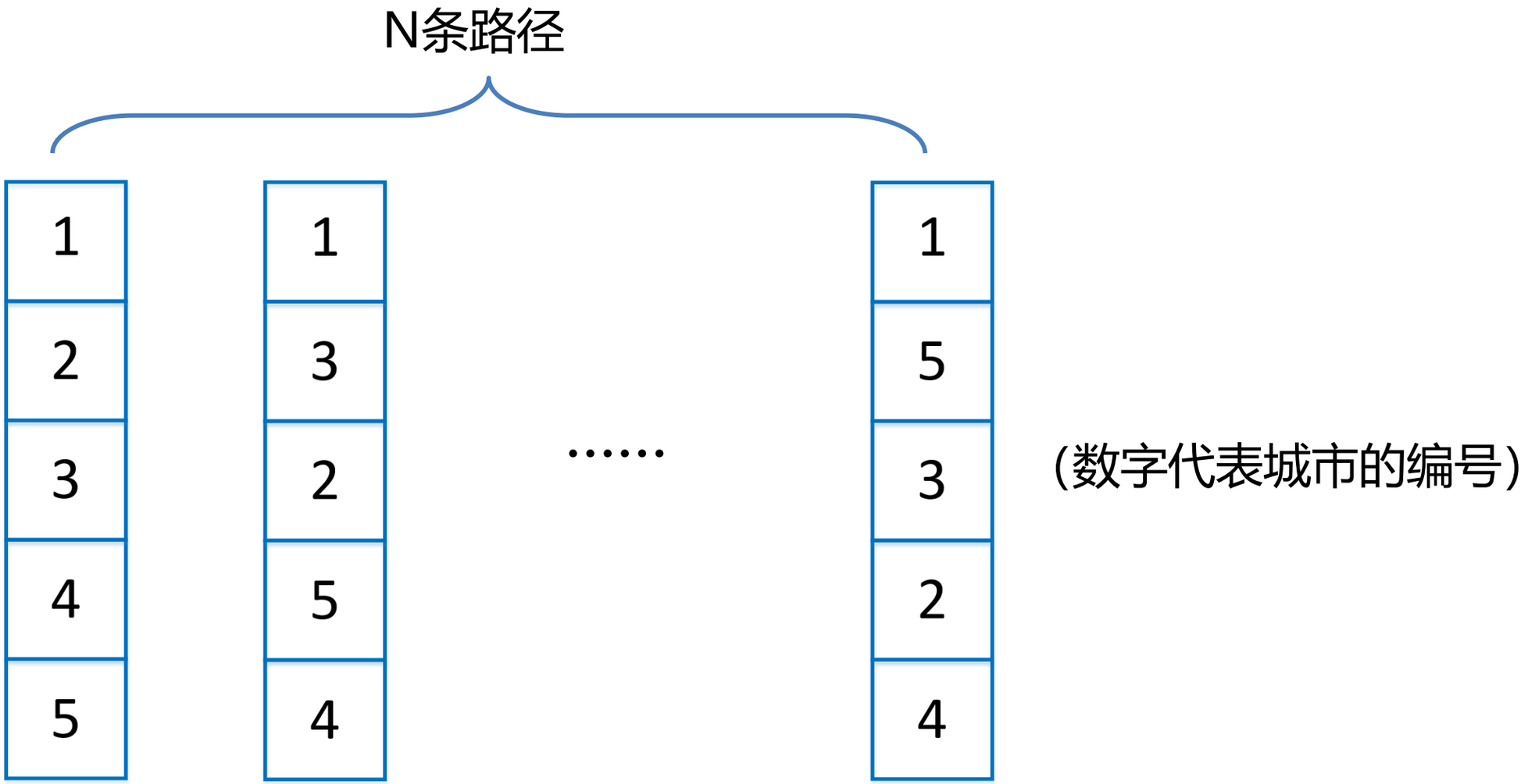
- **旅行商问题（Traveling Salesman Problem, TSP）**
  - 假设有一个旅行商人要拜访N个城市；
  - 需要规划一条路径，满足：每个城市只能拜访一次，且最后回到出发的城市；
  - 目标：求得路径的长度在所有路径中最小。





遗传算法 (Genetic Algorithm, GA)

- **第一步：**假设有5个城市，随机生成N条路径（群体，population）



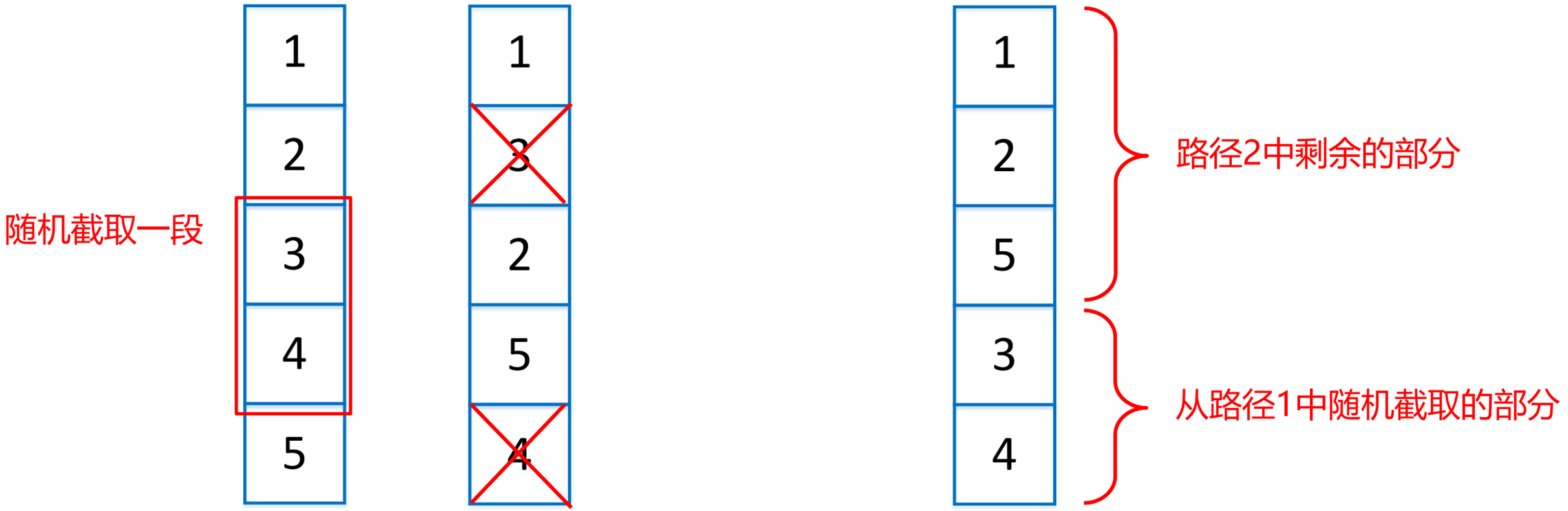
遗传算法 (Genetic Algorithm, GA)

- **第一步**：假设有city_num个城市，随机生成path_num条路径

```
def random_paths(city_num, path_num):
    paths = []
    path = [i for i in range(city_num)]
    for i in range(path_num):
        # 使用random.shuffle()随机打乱城市顺序
        random.shuffle(path)
        temp = path.copy()
        paths.append(temp)
    return paths
```

遗传算法 (Genetic Algorithm, GA)

- **第二步：**对每条路径概率性地使用交叉（crossover）生成新的路径



遗传算法 (Genetic Algorithm, GA)

- **第二步**：对每条路径概率性地使用交叉（crossover）生成新的路径

```
def crossover(paths, cross_rate):
    # 保存变异结果
    result = []
    path_num = len(paths)
    for i in range(len(paths)):
        # 0-1之间的随机数
        rand_num = np.random.rand()
        # 不变异
        if rand_num >= cross_rate:
            temp = paths[i].copy()
            result.append(temp)
```

遗传算法 (Genetic Algorithm, GA)

```
# 变异
if rand_num < cross_rate:
    list1 = paths[i].copy()
    rand_idx = random.randint(0, path_num - 1)
    while rand_idx == i:
        rand_idx = random.randint(0, path_num - 1)
    list2 = paths[rand_idx].copy() # 随机选择除list1以外的路径

    k1 = random.randint(0, len(list1) - 1)
    k2 = random.randint(0, len(list2) - 1)
    # 随机选择2个截取点
    while True:
        if k1 != k2:
            break
        k1 = random.randint(0, len(list1) - 1)
        k2 = random.randint(0, len(list2) - 1)
    if k1 > k2:
        k1, k2 = k2, k1
```

遗传算法 (Genetic Algorithm, GA)

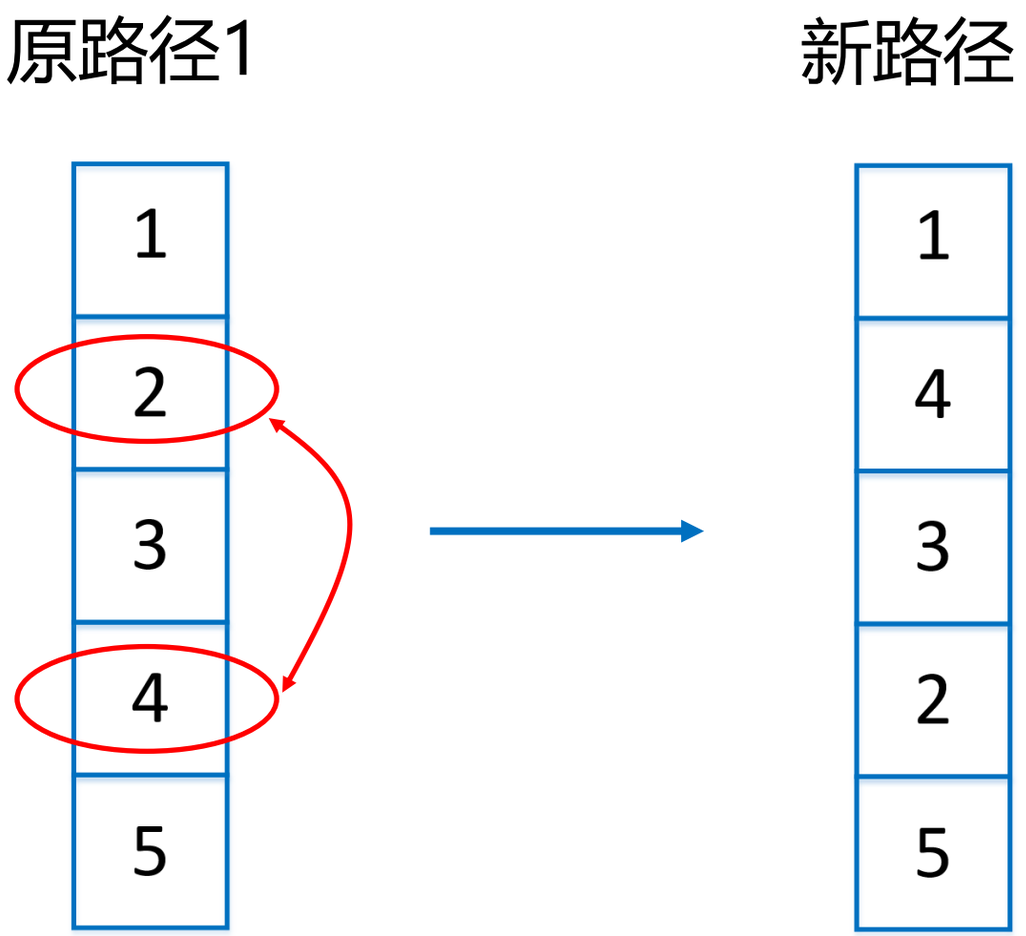
```
            # 截取第i条路径的一段
            segment1 = list1[k1:(k2 + 1)]
            segment2 = []
            for city in list2:
                if city not in segment1:
                    segment2.append(city)
            # 拼接
            temp = segment2 + segment1

            result.append(temp)

    return result
```

遗传算法 (Genetic Algorithm, GA)

- **第三步：**对每条路径概率性地使用变异（mutation）生成新的路径



遗传算法 (Genetic Algorithm, GA)

- **第三步**：对每条路径概率性地使用变异（mutation）生成新的路径

```
def mutation(paths, mutation_rate):
    path_length = len(paths[0])
    for path in paths:
        # 概率性变异
        if np.random.rand() < mutation_rate:
            mutate_point1 = np.random.randint(0, path_length)
            mutate_point2 = np.random.randint(0, path_length)
            # 2个变异点不能相同
            while mutate_point1 == mutate_point2:
                mutate_point2 = np.random.randint(0, path_length)
            path[mutate_point1], path[mutate_point2] = path[mutate_point2],
                                                       path[mutate_point1]

    return paths
```

遗传算法 (Genetic Algorithm, GA)

- **第四步：**计算N条新路径的长度，并使用轮盘赌选择（roulette wheel selection）选出N条路径
- **赌轮选择：**有放回抽取，长度越短的路径越容易被选中
  - ✓ 假设共有3条新路径：路径1长度为15km，路径2长度为30km，路径3长度为60km
  - ✓ 适应度（fitness）：路径1为 $\frac{1}{15}$，路径2长度为 $\frac{1}{30}$，路径3长度为 $\frac{1}{60}$
  - ✓ 选择N次，每次路径1被选中的概率均为：$\frac{\frac{1}{15}}{\frac{1}{15}+\frac{1}{30}+\frac{1}{60}}=\frac{4}{7}$
  - ✓ 同理，每次路径2被选中的概率为 $\frac{2}{7}$，路径3被选中的概率为 $\frac{1}{7}$

遗传算法 (Genetic Algorithm, GA)

- **第四步**：计算路径的适应度 (fitness)

```
def get_fitness(paths, coordinates):
    result = []
    for path in paths:
        result.append(1/(distance(path, coordinates)))
    # return result
    # 降低距离较长路径的适应度！
    return result - np.min(result)
```

遗传算法 (Genetic Algorithm, GA)

- **第四步**：计算路径的距离

```
def distance(path, coordinates):
    result = 0
    for i in range(len(path) - 1):
        city1 = coordinates[path[i]]
        city2 = coordinates[path[i + 1]]
        result = result + ((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)**0.5

    # 回到出发的城市
    city1 = coordinates[path[-1]]
    city2 = coordinates[path[0]]
    return result + ((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)**0.5
```

遗传算法 (Genetic Algorithm, GA)

- **第四步**：轮盘赌选择

```
def select(paths, fitness):
    fitness_sum = fitness.sum()
    # 轮盘赌选择
    # size：选择size次
    # replace：True代表有放回抽取
    # p：各元素的抽取概率
    temp = np.random.choice(np.arange(len(paths)),
                            size=len(paths), replace=True,
                            p=(fitness / fitness_sum))
    result = []
    for i in temp:
        result.append(paths[i])
    return result
```

遗传算法 (Genetic Algorithm, GA)

- **第五步：重复EPOCH次上述操作，然后返回当前最优解**

```
# 超参数
PATH_NUM = 200
CITY_NUM = 20
# 随机生成城市的坐标
city_coordinates = 100 * np.random.rand(CITY_NUM, 2)
CROSS_RATE = 0.6
MUTA_RATE = 0.2
epoch = 1000
```

遗传算法 (Genetic Algorithm, GA)

```
paths = random_paths(CITY_NUM, PATH_NUM)

best_distance = []
for i in range(epoch):
    paths = crossover(paths, CROSS_RATE)
    paths = mutation(paths, MUTA_RATE)
    fitness = get_fitness(paths, city_coordinates)

    # 当前迭代中，最优路径的长度
    best_path_idx = np.argmax(fitness)
    best_distance.append(distance(paths[best_path_idx], city_coordinates))

    paths = select(paths, fitness)
```

遗传算法 (Genetic Algorithm, GA)

```
fitness = get_fitness(paths, city_coordinates)
max_fitness_idx = np.argmax(fitness)

#打印结果
print('最优路径：', paths[max_fitness_idx])
print('最优路径的长度：', distance(paths[max_fitness_idx],
                                  city_coordinates))
print('每次迭代中的最优路径长度：', best_distance)
```

结束语

湖北大学
HUBEI UNIVERSITY

谢 谢!