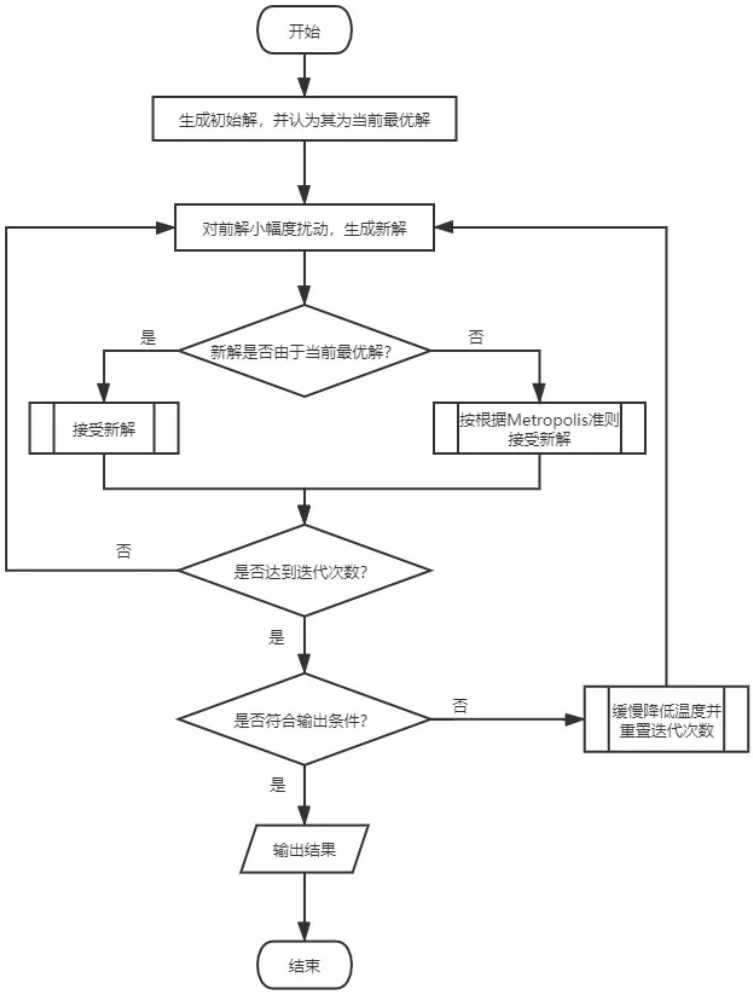
智能优化算法课程报告

彭程 2020011075

第一题：求解多极小函数

算法分析

此处我们采用模拟退火算法(SA),退火算法示意图如下：



其流程为：

1. 令 $T=T_0$ ，表示开始退火的初始温度，随机产生一个初始解 x_0 ，并计算对应的目标函数值 $E(x_0)$ ；
2. 令 $T=kT$, 其中 k 取值 0 到 1 之间, 为温度下降速率;
3. 对当前解 x_t 施加随机扰动, 在其邻域内产生一个新解 x_{t+1}, 并计算对应的目标函数值 $E(x_{t+1})$ ，计算

$$\Delta E=E(x_{t+1})-E(x_t)$$

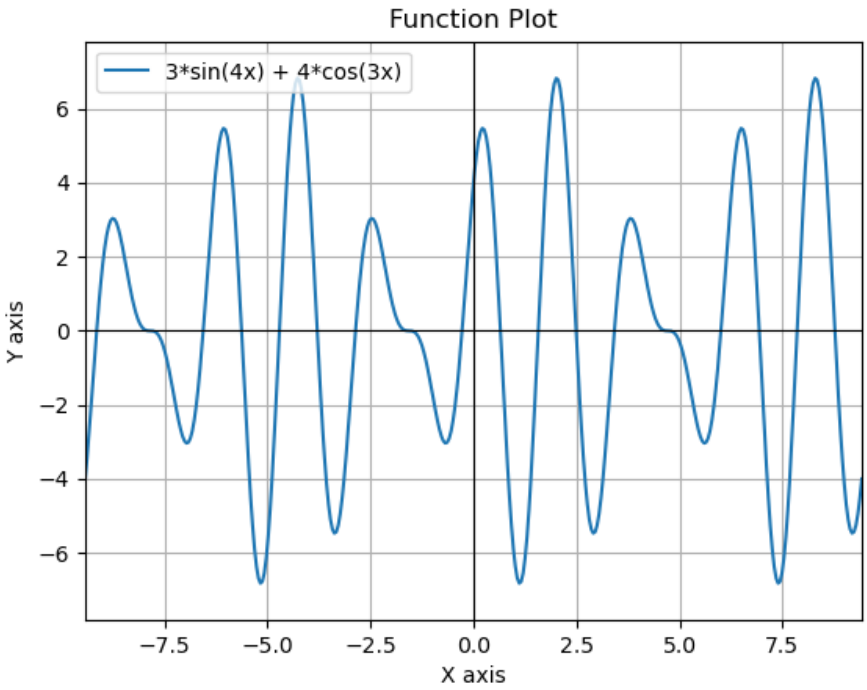
4. 若 $\Delta E<0$, 接受新解作为当前解, 否则按照概率 $e^{-\Delta E/kT}$ 判断是否接受新解;
5. 在温度 T 下, 重复 L 次扰动和接受过程, 即执行步骤 3 和 4 ;
6. 判断温度是否达到终止温度水平, 若是则终止算法, 否则返回步骤 2 .

问题求解

此处我们设定一个多极小函数：

$$f(x) = 3\sin(4x) + 4\cos(3x)$$

绘制其函数图像如下所示：



按照上述模拟退火算法编程求解:

选取如下的超参数：

```python
T_max = 10000 # Initial temperature
T_min = 0.001 # Final temperature
R = 0.9 # Cooling rate
num_iterations = 100 # Iterations per temperature
```

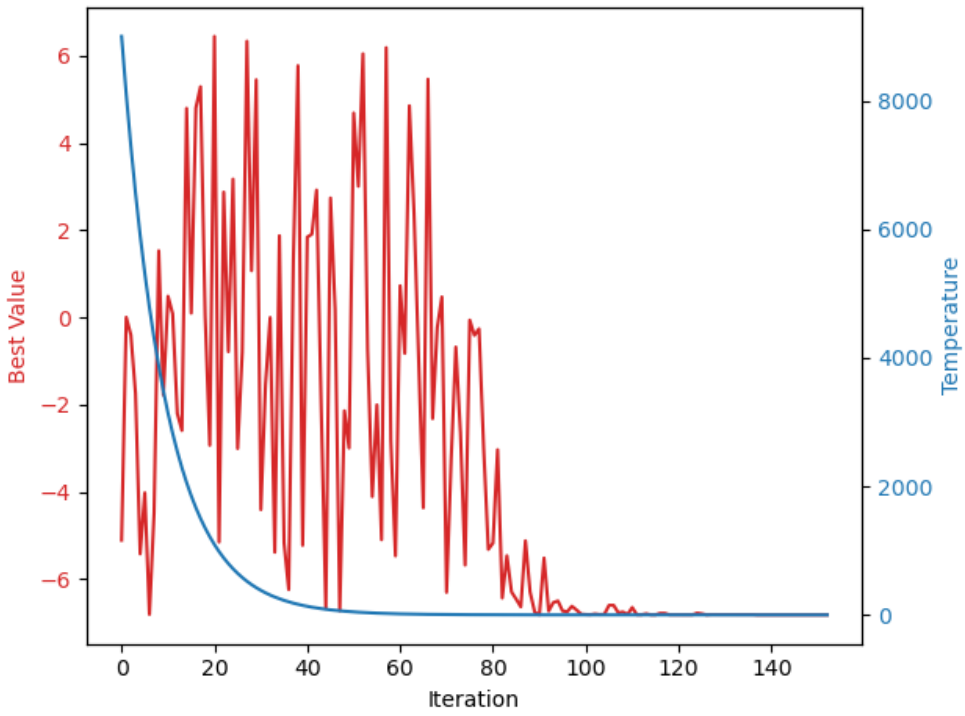
运行了20次，获得的结果如下：

```
min value:-6.82422 min x:1.12460
min value:-6.82441 min x:1.12348
min value:-6.82450 min x:1.12190
min value:-6.82416 min x:1.12487
min value:-6.82444 min x:1.12081
min value:-6.82259 min x:1.12881
min value:-6.82267 min x:1.12868
min value:-6.82296 min x:1.11587
min value:-6.82437 min x:1.12022
min value:-6.82377 min x:1.11779
min value:-6.82430 min x:1.11982
min value:-6.82425 min x:1.12444
min value:-6.81786 min x:1.13472
min value:-6.82363 min x:1.11740
min value:-6.82432 min x:1.12405
min value:-6.82399 min x:1.12552
min value:-6.82142 min x:1.11333
min value:-6.82340 min x:1.11682
min value:-6.82146 min x:1.11339
min value:-6.82431 min x:1.12410
```

其平均值，最优结果、最差结果，方差如下，

```
min value average:-6.82335
min value best:-6.82450
min value worst:-6.81786
min value variance:0.0000024448
```

随机选择其中一次变化情况展示如下：



函数优化中的特点

在**函数优化**问题中，如求解多峰值函数的最小值，解空间通常是连续的。

1. **连续解空间**：函数优化问题的解空间是连续的，这要求算法能够在连续域内有效搜索。
2. **求解策略**：SA算法在这类问题中通过生成连续的候选解并根据其函数值来接受或拒绝新解。
3. **平滑性和导数信息**：对于某些函数优化问题，解的平滑性或可导性可能会影响算法的性能。SA算法不依赖于导数信息，这使其适用于非平滑或噪声较多的优化问题。
4. **调参和效率**：函数优化问题可能需要算法参数（如初始温度、冷却速率等）的仔细调整，以达到较好的平衡点，从而在探索（寻找全局最优）和开发（优化当前解）之间取得平衡。

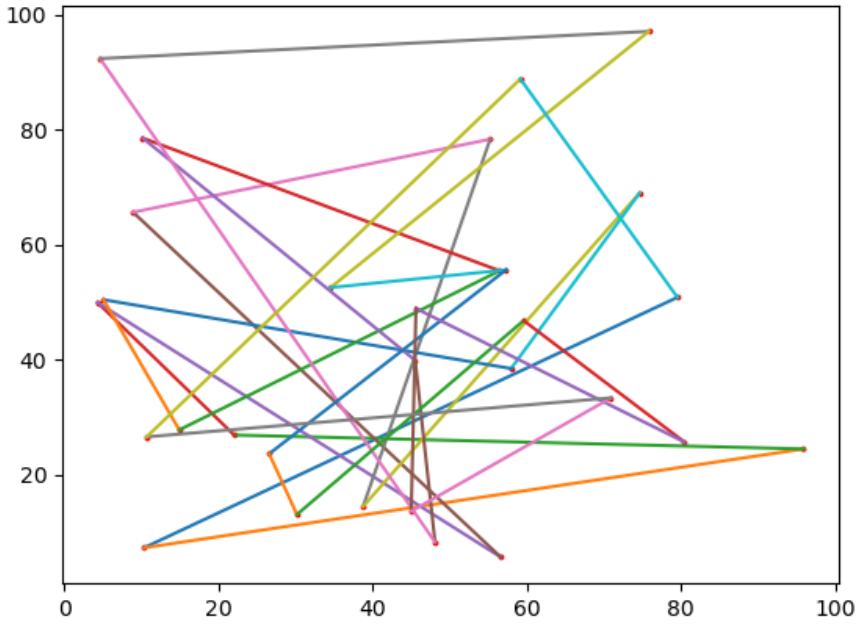
实验体会

使用模拟退火算法解决多极小值问题需要精确地调整参数，如初始温度、降温速率和终止条件，以确保算法不仅能找到局部最小值，而且能有效地逃离局部最优并探索更广泛的解空间。其中选择较高的初始温度，较慢的速率和较低的终止温度可以提高最后解的质量，不过也会相应提升求解时间。

第二题： 求解TSP问题

问题设定

首先初始化一张30个点的TSP问题



运行结果

同理第一题，按照上述模拟退火算法编程求解:
选取如下的超参数:

```python
T_max = T0# Initial temperature
T_min = 0.01 # Final temperature
R = 0.9 # Cooling rate
num_iterations = 200 # Iterations per temperature
```

其中T0设置具有一定随机性：

$$T0 = \frac{\Delta E}{-\log(p)}$$

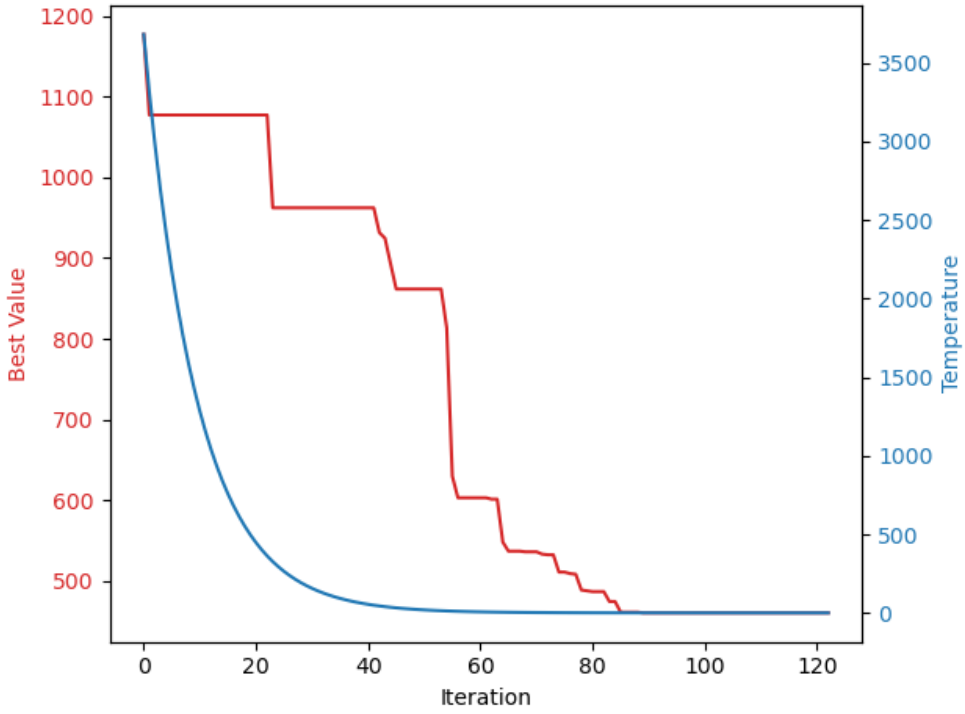
运行了20次实验结果如下：

```python
min value:555.39633
min value:550.55705
min value:490.86007
min value:474.91606
min value:605.00967
min value:471.93021
min value:527.61398
min value:518.93790
min value:518.72355
min value:500.30305
min value:511.86920
min value:527.24534
min value:539.11102
min value:502.82254
min value:468.57839
min value:540.75762
min value:460.22806
min value:471.97791
min value:519.58807
min value:461.18730
```

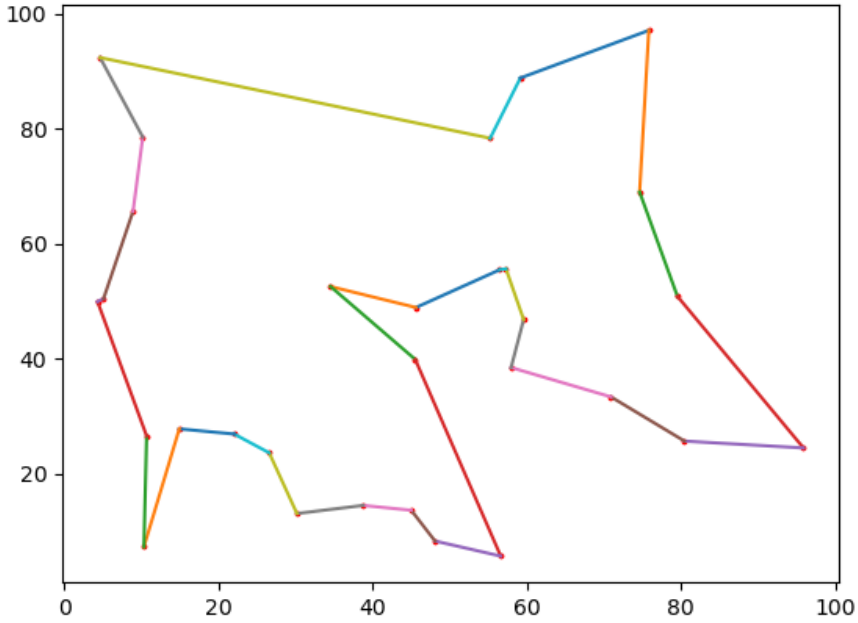
其平均值，最优结果、最差结果，方差如下，

```python
min value average:510.88
min value best:460.23
min value worst:605.01
min value variance:1329.15
```

选取一次目标函数变化曲线如下：



最优结果展示如下：



组合优化中的特点

在**组合优化**问题中，如旅行商问题（TSP），解空间通常是离散的。

1. **解空间和邻域结构**：这些问题的解空间通常是离散的，并且需要明确定义“邻域”的概念。邻域是指当前解可以通过小的、定义良好的变化所到达的解的集合。
2. **状态转移和接受准则**：SA在这些问题中通过一定的规则从一个解跳到另一个解，并根据退火原理接受或拒绝新的解。
3. **多样性和避免局部最优**：SA算法有助于避免局部最优解，因为它允许在早期阶段以一定概率接受劣解，从而保持解空间的探索多样性。
4. **计算复杂度**：对于一些复杂的组合问题，SA算法可能需要很长时间才能找到接近最优的解，特别是对于解空间非常大的问题。

实验体会

在解决TSP问题时，实验集中在如何有效地利用SA算法来找到近似最优解。于是我采取了多种不同的策略，例如初始温度的选取策略、随记路线的选取策略，还有不同的降温速率和终止条件，可以发现算法的效率和解的值还是很受这些参数的局限。SA的核心优势在于其能够跳出局部最优解并探索更广泛的解空间，但这也可能以牺牲计算效率为代价。正确调整算法参数对于在特定问题上实现最佳性能至关重要。