Intelligent Optimization Algorithms and Its Applications: Final Project (2021 Fall)

2019011430 徐赫临 自94

PA 1

利用至少一种智能算法求解多极小函数：

采用SA、GA、TS、PSO、DE或混合算法等

函数可自我选择

给出20次随机实验的统计结果（平均性能、最佳性能、最差性能、方差等）

给出典型的某次仿真过程中目标函数的变化曲线

讨论所用算法在函数优化中的特点

阐述实验体会

SA算法实现

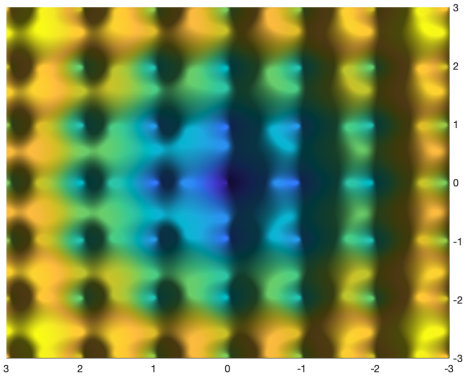
```
# simulated annealing algorithm
def simulated_annealing(objective, bounds, n_iterations, step_size, temp):
    # generate an initial point
    best = bounds[:, 0] + rand(len(bounds)) * (bounds[:, 1] - bounds[:, 0])
    # evaluate the initial point
    best_eval = objective(best)
    # current working solution
    curr, curr_eval = best, best_eval
    scores = list()
    # run the algorithm
    for i in range(n_iterations):
        # take a step
        candidate = curr + randn(len(bounds)) * step_size
        # evaluate candidate point
        candidate_eval = objective(candidate)
        # check for new best solution
        if candidate_eval < best_eval:
            # store new best point
```

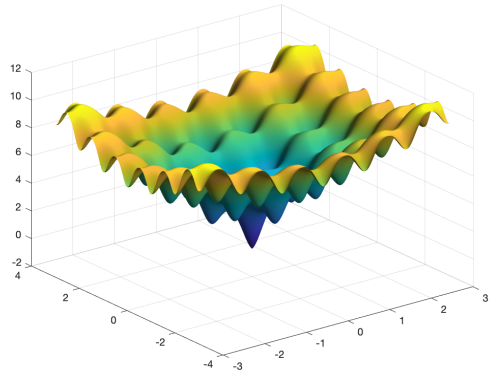
```
            best, best_eval = candidate, candidate_eval
            # keep track of scores
            scores.append(best_eval)
            # report progress
            print('>%d f(%s) = %.5f' % (i, best, best_eval))
        # difference between candidate and current point evaluation
        diff = candidate_eval - curr_eval
        # calculate temperature for current epoch
        t = temp / float(i + 1)
        # calculate metropolis acceptance criterion
        metropolis = exp(-diff / t)
        # check if we should keep the new point
        if diff < 0 or rand() < metropolis:
            # store the new current point
            curr, curr_eval = candidate, candidate_eval
    return [best, best_eval, scores]
```

2d simulated annealing

函数选择

2d实验部分选择了常见的测试优化算法的ackley函数





$$f(\mathbf{x}) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

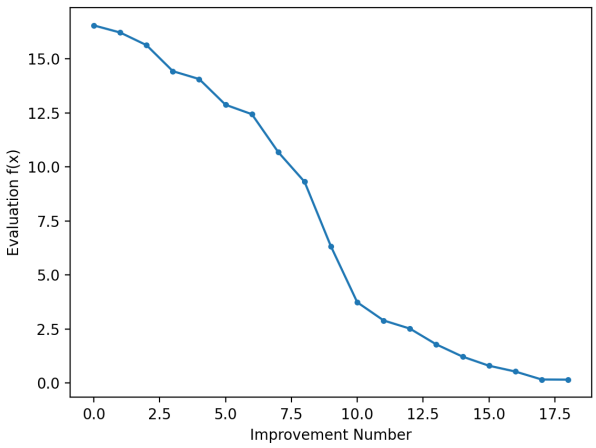
20次随机实验结果

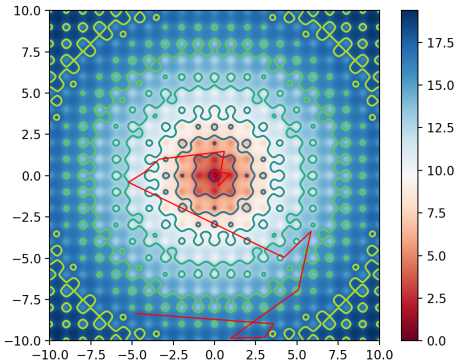
```
20 time resluts: [0.020182200826834418, 0.032351417428042595, 0.03583542712059895, 0.0
53441924713528266, 0.05407641462101154, 0.05506287805685517, 0.05731397304750985, 0.06
832561101478163, 0.08372007541857229, 0.09365603640205578, 0.09473863552927142, 0.1007
```

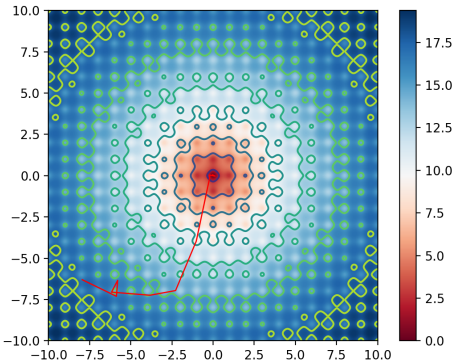
```
0310547131811, 0.10235594519278379, 0.1047953691892296, 0.1156250981996898, 0.11850602
164392399, 0.17689258714859335, 0.23108421313588323, 0.25483555956498805, 0.2631665863
2154244]
mean: 0.10583345400235071, var: 0.005165610857730473
```

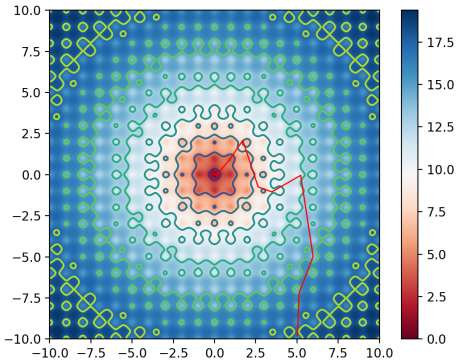
- 理论最小值：0
- 实验最小值：0.020182200826834418
- 实验最差情形：0.26316658632154244
- 平均值：0.10583345400235071
- 方差：0.005165610857730473

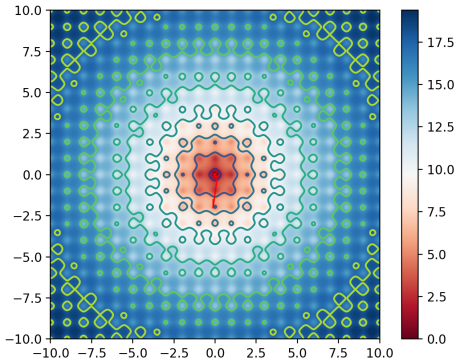
典型实验结果





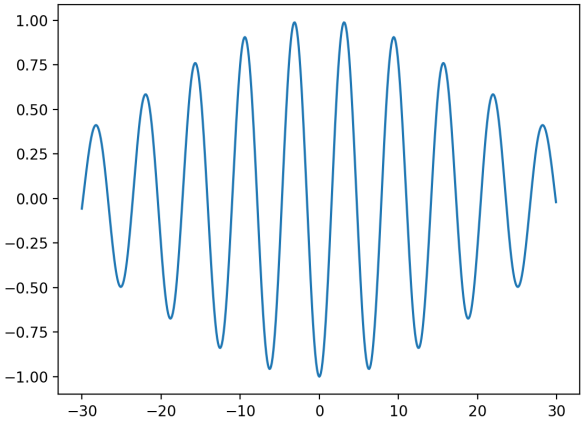






1d simulated annealing

函数选择




```
def objective(x):
    return -np.exp(-(x/30.)**2.0) * np.cos(x)
```

$$y = -e^{-\frac{x^2}{900}}\cos(x), 30 \leq x \leq 30$$

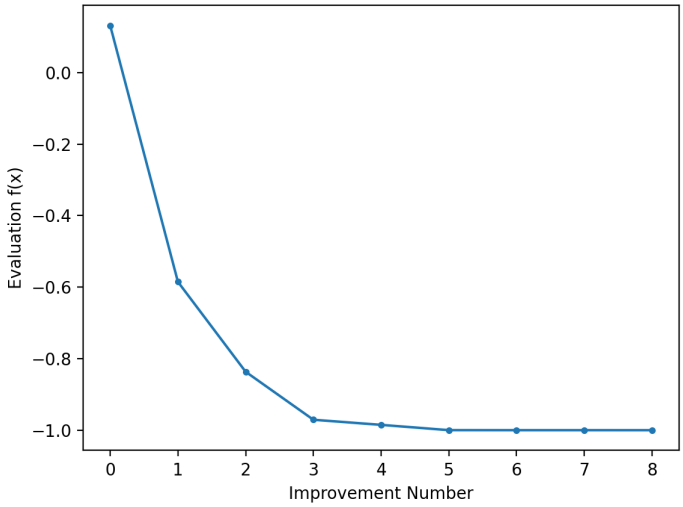
20次随机实验结果

```
20 time resluts: [-0.9999999954768114, -0.9999999876953939, -0.9999999161536416, -0.99
99999816663443, -0.9999995408417922, -0.9999985719171752, -0.9999985466020315, -0.9999
984532959044, -0.9999978442634686, -0.9999977083645887, -0.9999965979104375, -0.999995
8318189385, -0.9999956370927637, -0.999994238190816, -0.9999934942868074, -0.999982082
0721179, -0.9999779989887283, -0.9999165717740044, -0.8384424873519079, -0.44543964455
```

```
56276]
mean: -0.9641862480159651, var: 0.016209764684511328
```

- 理论最小值：-1
- 实验最小值：-0.9999999954768114
- 实验最差情形：-0.4454396445556276
- 平均值：-0.9641862480159651
- 方差：0.016209764684511328

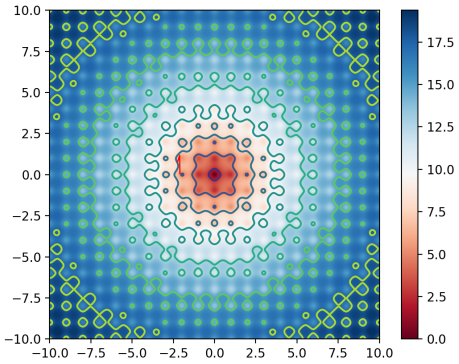
典型实验结果目标函数的变化曲线

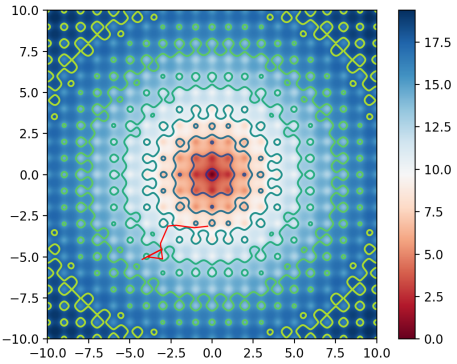


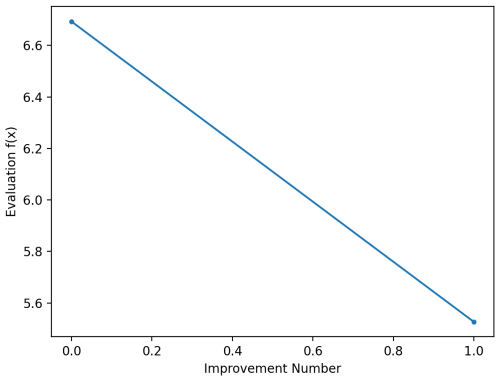
讨论所用算法在函数优化中的特点

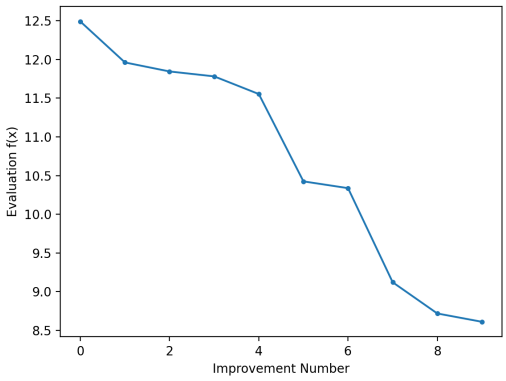
模拟退火算法有两个重要的参数，影响其性能：

1）初始温度：初始温度影响着在一开始，搜索范围跳跃出局部最小值的能力。如果初始温度过低，则一开始搜索点就不具有跳出局部最小值的能力，搜索容易陷入局部最小值。如果初始温度过高，则在有限次迭代之后，温度还是过高，算法总是大概率接受更差的解，导致算法无法收敛。





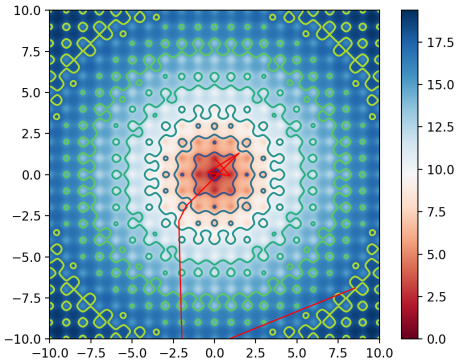


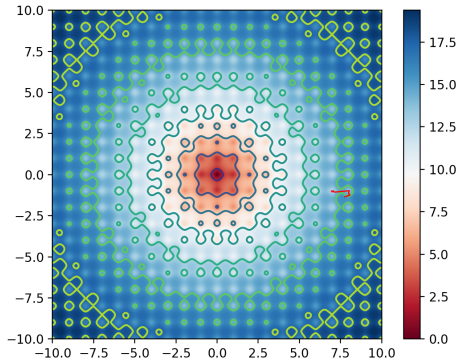


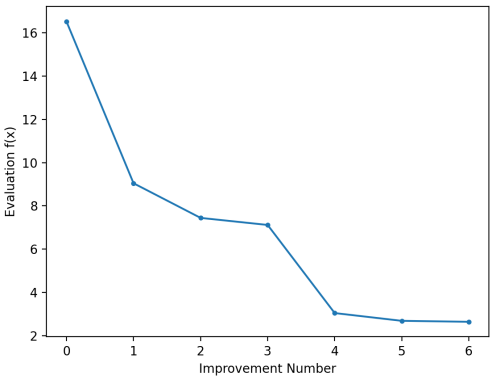
温度过高：类似于随机游走，找不到更优点，无法迭代收敛，因此可见迭代点数目很少

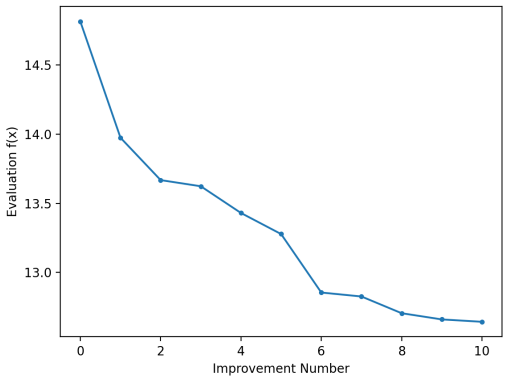
温度过低：每次总是很严格地寻找更优点才迭代，陷入局部极小值

2）搜索步长：搜索步长表征每一次随机搜索时，在多大范围内寻找新的解。搜索步长过小，一旦小于局部极小值点“凹陷”的直径，则无法搜索到局部极小值之外范围的点，自然也难以跳出局部极小值。搜索步长过大，则每次随机选取的点过于随机，不利于在合理范围内跳出局部极小值，难以收敛。









搜索步长过大：在过大范围内随机选新点，不利于收敛，最终收敛也不够精确

搜索步长过小：在很小范围内随机选点，无法跳出局部极小值

实验体会

对于每一种算法，不同参数调整具有不同物理意义，需要根据具体问题+对应的物理意义调整参数，而不要盲目乱调参数。在理解参数意义的基础上调参数，才能取得鲁棒的好结果。

PA2

利用至少一种智能优化算法求解TSP：

采用SA、GA、TS、PSO、DE、HNN或混合算法等

规模可自我选择

给出20次随机实验的统计结果（平均性能、最佳性能、最差性能、方差等）

给出典型的某次仿真过程中目标函数的变化曲线

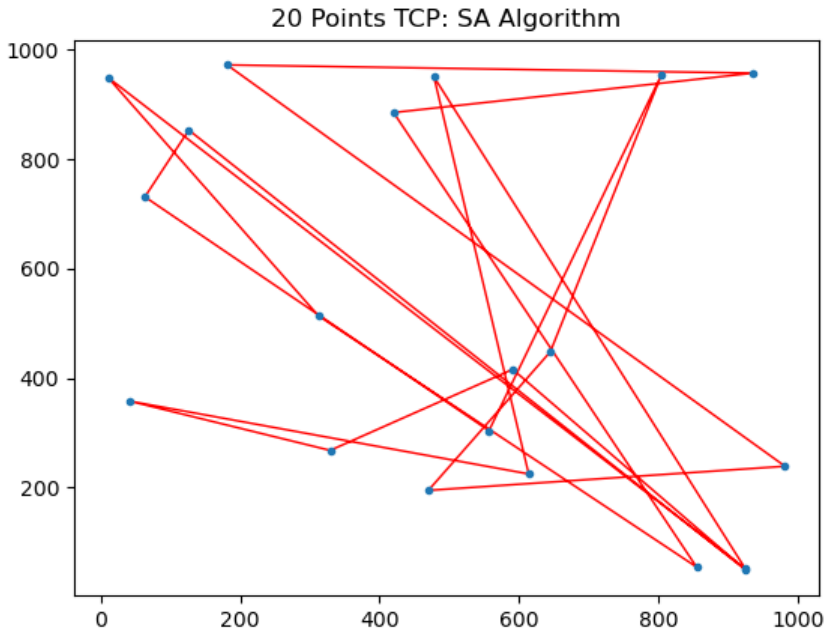
给出最优结果的图形

讨论所用算法在组合优化中的特点

阐述实验体会

20点旅行商问题：SA求解方法

问题设置：



我随机生成了20个点，并制定了随机点初始路径，如上图。

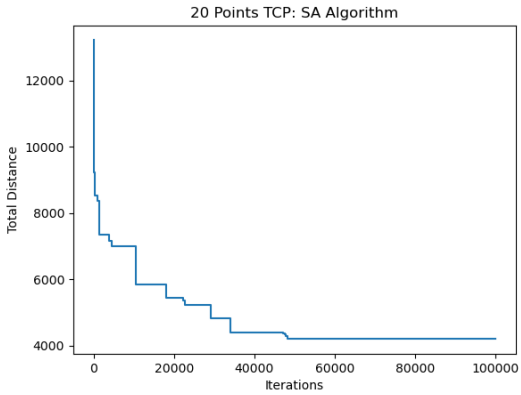
20次随机实验结果

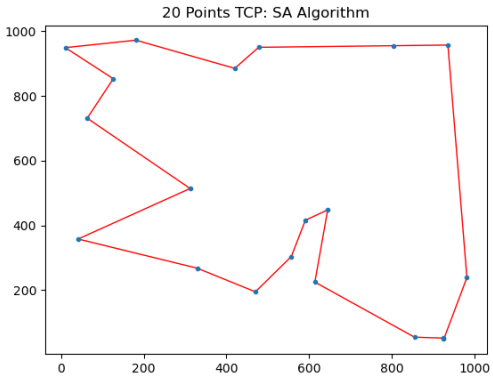
```
num_points: 20, iterations: 100000
temperature: 5000000, width: 1000
20 time resluts: [4180.364716887144, 4180.364716887145, 4180.364716887145, 4180.364716
887145, 4180.364716887146, 4181.762449158431, 4181.762449158432, 4193.550256504409, 41
99.313337171338, 4200.711069442627, 4212.498876788602, 4227.36192943267, 4231.62705007
3781, 4243.414857419759, 4264.285874201839, 4272.816441064759, 4301.295339604637, 430
7.977552380077, 4319.765359726054, 4353.577032654315]
mean: 4229.677172960873, var: 3057.646926433116
```

- 理论最短路径：4180.364716887144

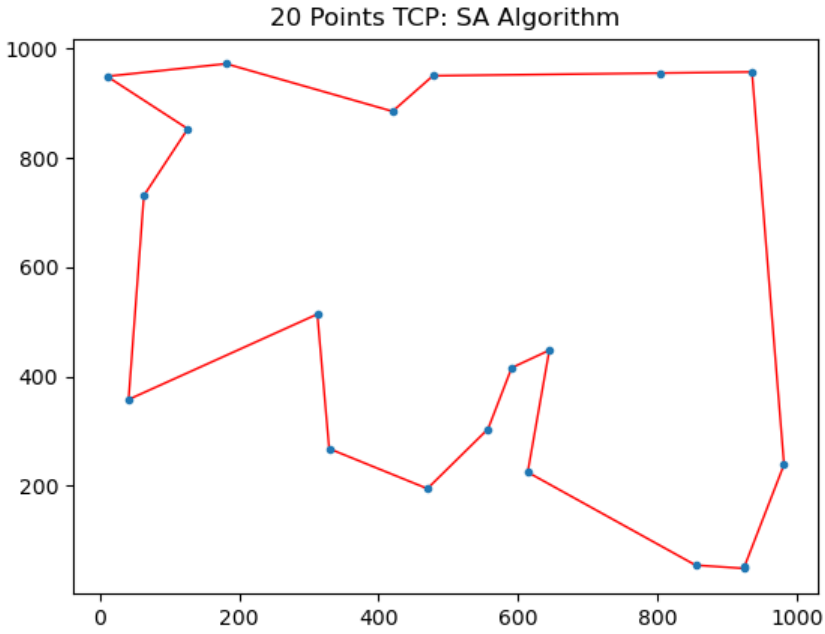
- 最佳性能：4180.364716887144
- 最差性能：4353.577032654315
- 平均性能：4229.677172960873
- 方差：3057.646926433116

典型实验结果变化曲线





最优图形



讨论所用算法在函数优化中的特点

模拟退火算法有两个重要的参数，影响其性能：

1）初始温度：初始温度影响着在一开始，搜索范围跳跃出局部最小值的能力。如果初始温度过低，则一开始搜索点就不具有跳出局部最小值的能力，搜索容易陷入局部最小值。如果初始温度过高，则在有限次迭代之后，温度还是过高，算法总是大概率接受更差的解，导致算法无法收敛。

2）迭代数量：搜索迭代数量直接决定着搜索能遍历的数目，迭代数量越多，效果越好。但是，如果初始温度过低，则一味增加迭代次数也无法导致跳出局部最小值。

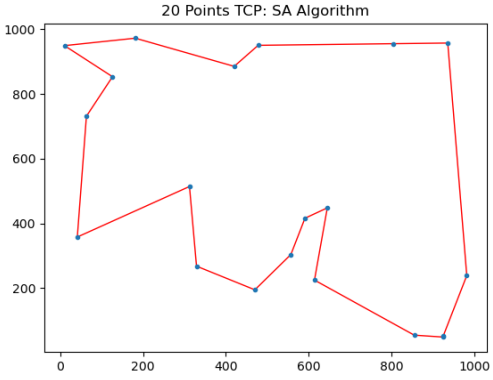
以下是多次试验的结果：

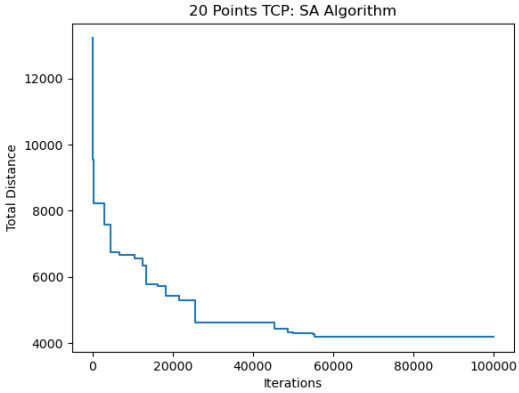
```
num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4225.964197161384, 4272.816441064757, 4289.325885867151, 4333.709280
261613, 4337.379136025078, 4340.588219053788, 4347.769211099766, 4349.807694958079, 43
53.577032654315, 4381.851151136022, 4414.598582076289, 4418.336453631372, 4477.9857337
55672, 4515.924516090804, 4607.778835430022, 4624.799838445881, 4639.650696543058, 469
2.453984504474, 4943.074593407262, 5107.785840626364]
mean: 4483.758866189657, var: 52137.29134028975

num_points: 20, iterations: 10000
temperature: 50000, width: 1000
20 time resluts: [4243.414857419759, 4301.113693213127, 4320.406446128418, 4452.604343
195991, 4528.185617532759, 4567.292130723923, 4586.625735253013, 4587.041904450578, 46
36.616985274928, 4661.210670854275, 4771.291372859532, 4804.914273522584, 4844.3048952
31541, 4844.977447646992, 4870.239913485527, 5055.745351858953, 5140.42375482988, 515
8.58118747058, 5428.223297258952, 5542.084543535437]
mean: 4767.264921087337, var: 127252.90248981921

num_points: 20, iterations: 10000
temperature: 5000, width: 1000
20 time resluts: [4415.792245734792, 4453.5970043002035, 4568.819279036222, 4599.92211
966362, 4674.247665500214, 4816.249766731406, 4912.906478403182, 4927.706870389107, 51
63.351136165838, 5196.436650221846, 5212.559680912331, 5255.036296700187, 5256.9004795
37719, 5268.165173777041, 5327.587540630577, 5456.205453763274, 5475.111758810169, 548
7.615887317174, 5502.136084743797, 5878.080862100481]
mean: 5092.421421721959, var: 162273.7147671089

num_points: 20, iterations: 10000
temperature: 5000000, width: 1000
20 time resluts: [5550.324485816736, 6109.254153272474, 6280.10853254259, 6290.7284584
78821, 6363.838907178741, 6368.640311045622, 6412.3317447445625, 6412.402315764731, 64
34.388397029592, 6556.32974354447, 6592.726062519265, 6600.107312745969, 6700.02041139
7398, 6759.18148071583, 6784.500215750019, 6803.447632982727, 6871.339532632184, 6922.
888849241585, 7062.5446834175755, 7073.078696417142]
mean: 6547.409096361902, var: 127411.57251486719

num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4227.361929432669, 4255.706458534543, 4257.925674375615, 4278.624262
619308, 4289.325885867151, 4308.274506151345, 4341.985951325076, 4366.762572271579, 43
71.98032748821, 4380.902075151613, 4417.761454153959, 4466.44204842897, 4478.525049066
667, 4493.265591106241, 4511.256786011476, 4746.126383378082, 4826.768531480086, 4880.
```

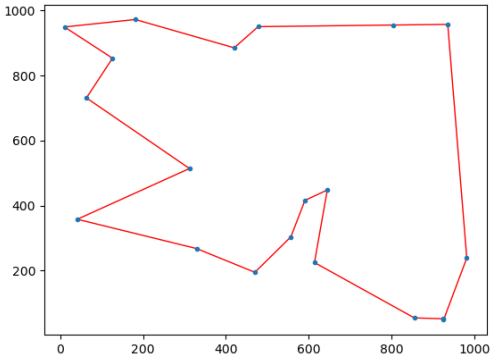
```
049230579765, 5054.369892308043, 5106.396219029942]
mean: 4502.990541438017, var: 73222.82250796437

num_points: 20, iterations: 1000
temperature: 500000, width: 1000
20 time resluts: [6322.894828434323, 6362.447117390229, 6428.280377747009, 6904.308264
999223, 7006.292941053015, 7105.367314729384, 7163.886923849444, 7195.3210369141325, 7
199.864760915929, 7225.21476476232, 7372.445036627539, 7409.557641144087, 7417.5602853
96939, 7481.977605201222, 7691.748518895744, 7771.782267383136, 7798.2396326641, 7941.
4399857305125, 8115.144919683257, 8228.91497132513]
mean: 7307.134459742334, var: 291879.1644182534

num_points: 20, iterations: 1000
temperature: 50000, width: 1000
20 time resluts: [4443.496629746271, 4585.937898278523, 4853.42539796841, 4865.7531743
15851, 4900.227687651779, 4903.94619226106, 5010.85623147985, 5095.592776167704, 5124.
747138090074, 5161.598008583518, 5167.417091708816, 5200.39458734173, 5226.72136424406
7, 5260.006462533211, 5338.58371380379, 5347.718845689858, 5406.37235720962, 5588.1463
29110994, 5689.337414926801, 5843.599075405015]
mean: 5150.6939188258475, var: 118801.51906160491

num_points: 20, iterations: 1000
temperature: 5000, width: 1000
20 time resluts: [4264.416890779491, 4406.369157252724, 4593.419448887434, 4601.607983
9739305, 4662.030572042779, 4753.194105124194, 4753.905686505785, 4845.194395974138, 4
895.2651449111, 4920.785546956727, 4944.969107539146, 5130.271321007833, 5132.80901751
274, 5178.1121478284, 5238.285003149277, 5361.190599829219, 5511.637862838899, 5607.27
7373461088, 5696.868024674863, 5715.612215228378]
mean: 5010.661080273907, var: 177738.07505391404

num_points: 20, iterations: 1000
temperature: 500, width: 1000
20 time resluts: [4321.804178399705, 4656.062063606244, 4672.938238015503, 4677.189272
927086, 4678.673281286125, 4684.864074716219, 4949.871436814195, 5055.560068834609, 51
04.281830225827, 5139.40417009937, 5230.350111632784, 5305.771785644816, 5382.73973060
8207, 5392.763034451356, 5416.436140196874, 5640.594072000149, 5657.931720144342, 574
3.519511636273, 5802.930389091382, 6524.748123762433]
mean: 5201.921661704675, var: 273620.0403096674

num_points: 20, iterations: 100000
temperature: 50000, width: 1000
20 time resluts: [4181.762449158432, 4237.752004507359, 4321.804178399706, 4451.608050
999868, 4476.107296419981, 4579.286204796572, 4593.828067820963, 4598.081795939568, 47
09.309442064008, 4742.93209447533, 4789.00039603876, 4804.1881811022395, 4877.25956928
7918, 4927.594448304403, 4955.137804168864, 5021.305135180255, 5086.866614370446, 509
2.843678966114, 5287.537520072284, 5633.288131582315]
mean: 4768.374653182769, var: 130121.54048189071

num_points: 20, iterations: 100000
temperature: 500000, width: 1000
20 time resluts: [4181.762449158432, 4193.550256504408, 4199.313337171339, 4299.908206
537494, 4307.977552380077, 4309.672238422633, 4342.646882019861, 4354.974764925604, 43
65.364840000292, 4366.762572271579, 4382.978616299315, 4394.766423645291, 4398.1571530
65513, 4426.1610044392255, 4440.558546652015, 4486.158026926253, 4488.906926575163, 44
88.906926575163, 4504.435490342767, 4572.149952593406]
mean: 4375.255608325291, var: 11524.71121889219

num_points: 20, iterations: 100000
```

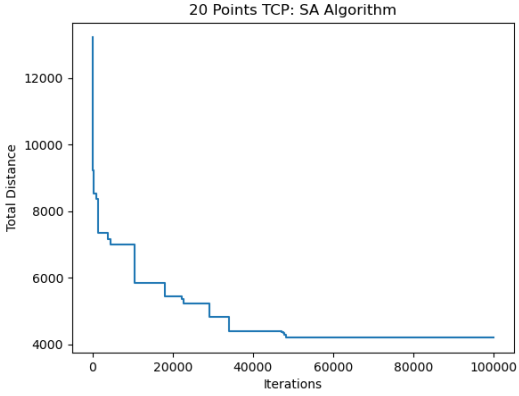
```
temperature: 5000000, width: 1000
20 time resluts: [4180.364716887144, 4180.364716887145, 4180.364716887145, 4180.364716
887145, 4180.364716887146, 4181.762449158431, 4181.762449158432, 4193.550256504409, 41
99.313337171338, 4200.711069442627, 4212.498876788602, 4227.36192943267, 4231.62705007
3781, 4243.414857419759, 4264.285874201839, 4272.816441064759, 4301.295339604637, 430
7.977552380077, 4319.765359726054, 4353.577032654315]
mean: 4229.677172960873, var: 3057.646926433116

num_points: 20, iterations: 100000
temperature: 50000000, width: 1000
20 time resluts: [5301.4664896306185, 5599.245613411695, 5722.485505287952, 5725.16828
3142677, 5732.612160392762, 5785.673115945651, 5821.279647942547, 5870.909595532691, 5
880.351201394075, 5892.496760467923, 5925.715387978934, 5954.269435841041, 5958.961878
379299, 6015.653361704952, 6040.52842487016, 6129.737956287892, 6164.072280997705, 628
0.227539455653, 6313.09857361098, 6323.023581208293]
mean: 5921.848839674175, var: 63510.181776834856

num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4192.152524233121, 4212.498876788602, 4239.149736778646, 4264.285874
201839, 4272.816441064757, 4288.109799987374, 4342.6633359158595, 4361.595502304056, 4
365.364840000293, 4368.072819000623, 4425.399500137881, 4434.251553283155, 4437.276035
7573125, 4442.624969562594, 4455.067929670909, 4496.9277261995785, 4498.937146343489,
 4678.5002617198, 4690.060790008936, 5132.089791480321]
mean: 4429.892272721957, var: 45544.61384776955

num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4212.498876788603, 4227.361929432671, 4271.41870879347, 4274.2305118
07695, 4308.274506151345, 4319.889630810096, 4325.006965622718, 4379.163791893699, 438
0.379877773476, 4434.771524169443, 4435.878303486026, 4450.210318728581, 4460.11863714
0482, 4465.561499824664, 4485.625711310992, 4503.930442018155, 4505.255197774935, 452
8.09438711505, 4552.268615280931, 4662.618496947117]
mean: 4409.1278966435075, var: 14085.467852546786

num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4271.41870879347, 4307.058420271568, 4345.497087607588, 4348.3844397
6422, 4349.807694958079, 4349.80769495808, 4351.159940519989, 4352.657900545806, 4370.
764241608434, 4424.381449094754, 4428.726528706061, 4441.227237291305, 4458.4428524212
41, 4463.4032945045265, 4473.735837501726, 4475.625674709096, 4477.985733755672, 4505.
833222614056, 4640.29798251728, 4890.1178193728665]
mean: 4436.316688075791, var: 18521.054241918253

num_points: 20, iterations: 10000
temperature: 500000, width: 1000
20 time resluts: [4181.762449158432, 4212.498876788603, 4227.361929432669, 4228.631031
533456, 4256.700624791552, 4266.096533609232, 4278.624262619307, 4289.507532258662, 43
17.018189067709, 4330.8590746738855, 4342.6468820198625, 4381.580884028028, 4424.64856
4860358, 4432.5704023314365, 4437.8341044776225, 4455.067929670908, 4503.930442018155,
4525.619481203257, 4637.916014482293, 4768.018148535332]
mean: 4374.944667878038, var: 22887.167769322616
```

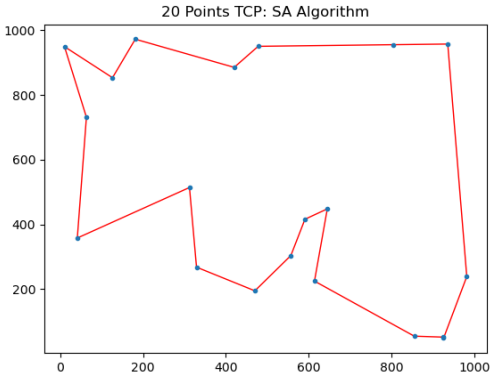
对应的图像如下

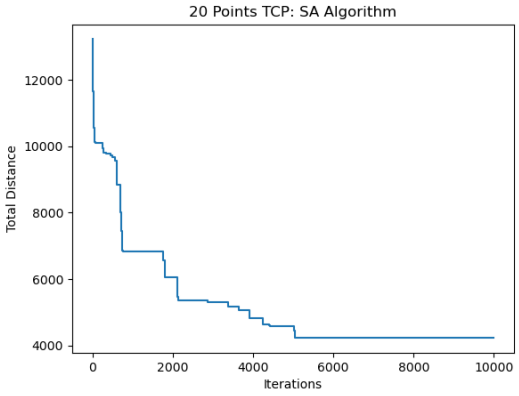

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

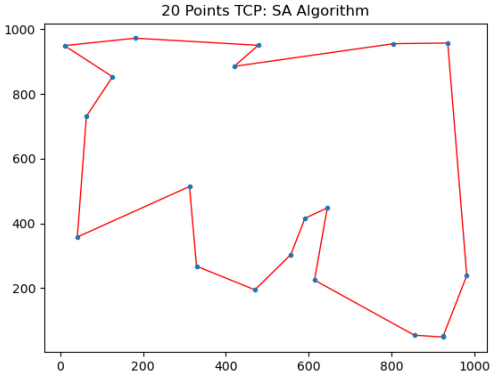

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
20000
40000
60000
80000
100000
Iterations

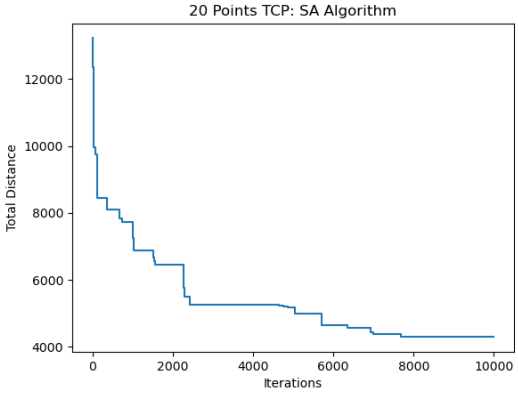

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

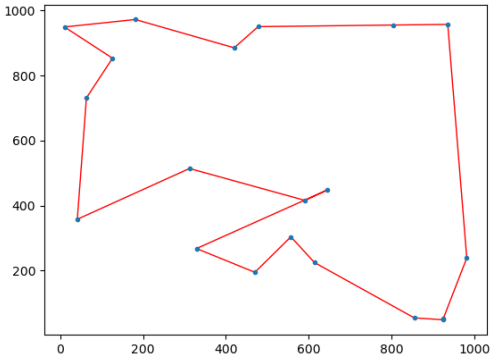

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
20000
40000
60000
80000
100000
Iterations

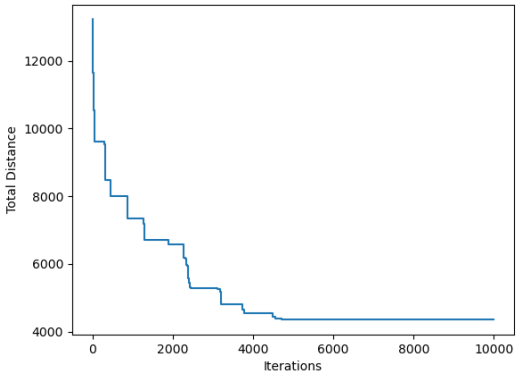

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

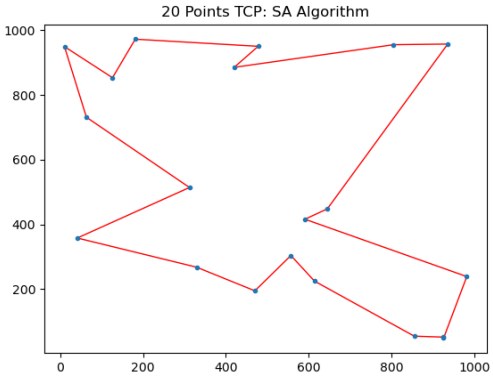

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
2000
4000
6000
8000
10000
Iterations

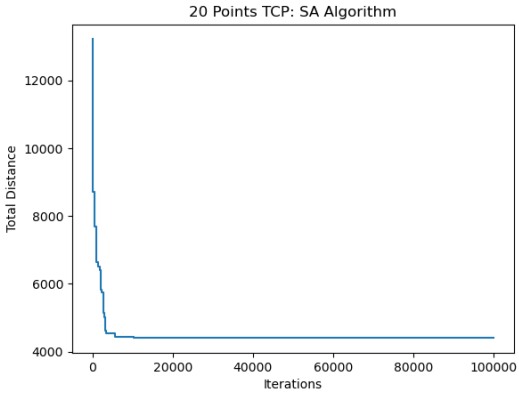

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

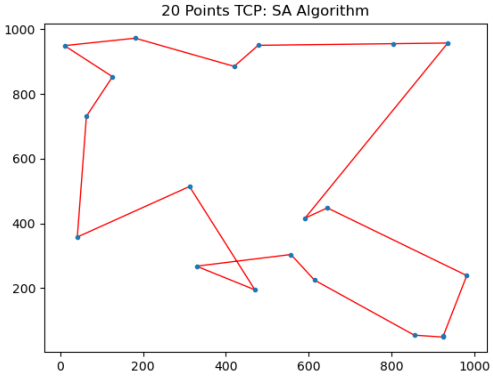

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
2000
4000
6000
8000
10000
Iterations

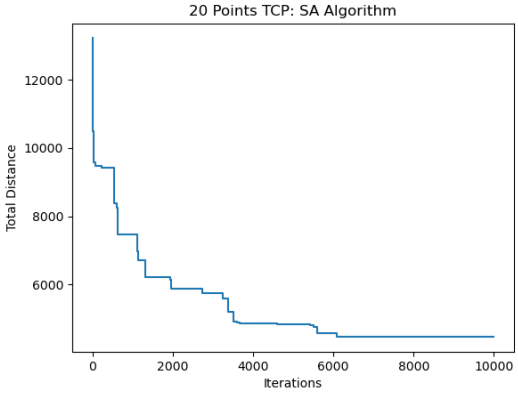

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

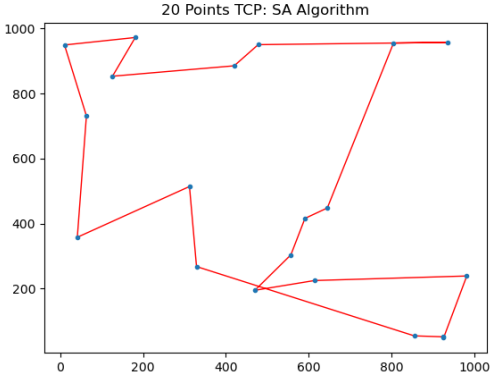

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
2000
4000
6000
8000
10000
Iterations

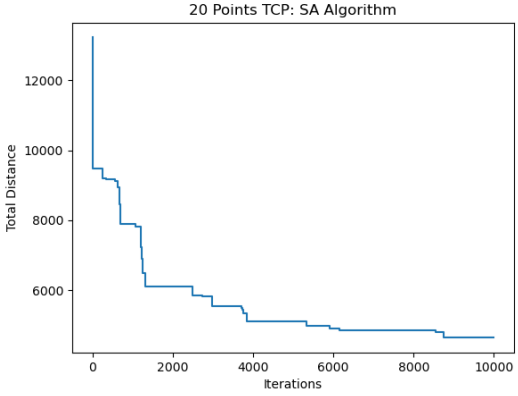

20 Points TCP: SA Algorithm
1000
800
600
400
200
0
200
400
600
800
1000

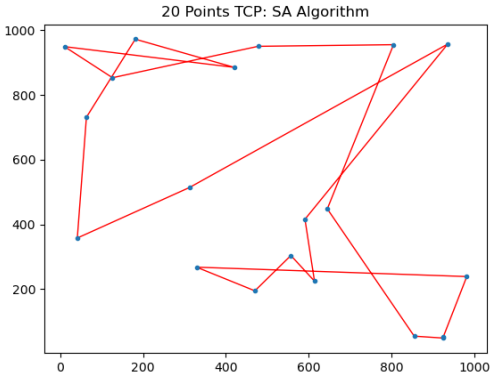

20 Points TCP: SA Algorithm
Total Distance
12000
10000
8000
6000
4000
0
20000
40000
60000
80000
100000
Iterations

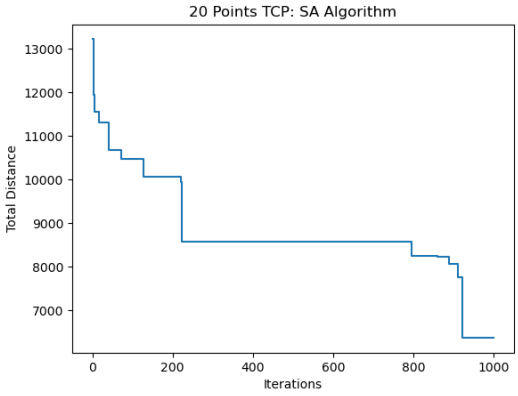












实验体会

对于每一种算法，不同参数调整具有不同物理意义，需要根据具体问题+对应的物理意义调整参数，而不要盲目乱调参数。在理解参数意义的基础上调参数，才能取得鲁棒的好结果。