旅行商问题的几种近似求解方法分析与对比

沈冠霖

（清华大学 软件学院，北京 100084）

摘　要：本文采用禁忌搜索算法、遗传算法和模拟退火算法三种算法求解旅行商问题，并且分析了三种算法相应参数对效果的影响，以及比较了三种算法的效果和计算耗时。一方面，三种算法内部的长度性参数选择需要适中，过小会导致搜索范围较小难以跳出局部最优解，过大会影响算法效率和效果。另一方面，最为复杂的遗传算法效果最好，也最慢。最为简单的模拟退火算法最快，但是效果最差。禁忌搜索算法效果位于两者之间，而耗时和遗传算法相当。在实际问题求解中，需要根据实际需求，权衡速度和效果，选择最合适的算法。

关键词：旅行商问题；禁忌搜索；遗传算法；模拟退火算法

1　问题和实验概述

旅行商问题（Travelling salesman problem, TSP）是组合优化领域的一个经典问题，由于其是 NP 难问题，因此解决此问题常常使用近似算法。本文使用了禁忌搜索算法、遗传算法和模拟退火算法进行求解，比较了各个算法参数对求解效果和计算效率造成的影响，并且比较了这几个算法的求解效果和计算效率。

本文使用的数据集是 Reinelt[1]提供的 TSPLIB 数据集中的 4 组数据，即 burma14、att48、eil101 和 a280。这四组数据的节点个数分别为 14、48、101 和 280，因为硬件性能限制，本文没有使用过于庞大的数据。

本文的代码全部在 Windows 10 操作系统下基于 python 语言进行实现。得到的具体结果，包括实验参数、最优解、最优值和运行时间等，可以在 results 文件夹下查看。由于这些都是随机化算法，为了保证实验可以复现，我事先固定了随机数种子，可以通过运行 src 文件夹中的 run_all.py 文件来复现实验结果。

2　禁忌搜索算法

2.1　算法介绍和实现方法

禁忌搜索算法（Tabu Search）是 Glover[2]在 1986 年提出的经典算法。这一算法基于局部搜索算法，引入了禁忌技术以防止重复前面搜索的结果，同时引入了特赦机制来保证能够搜索到较好的解。我的实现方式如下：

1) 初始化阶段，设节点个数为N，随机生成一个访问序列作为初始解和初始最优解，初始化禁忌表为一个最大长度为M的空队列。禁忌表采用双向禁忌机制，如果禁忌(i,j)的交换，则(j,i)的交换也会被同时禁忌。
2) 迭代阶段，使用 2-opt 方法随机选择邻域中的K个状态，将其按照边权之和从小到大排序。
3) 看这个状态序列的第一个元素是否比当前最优值更优。如果更优，更新当前的最优解和最优值为这一状态，跳转到 4，否则跳转到 6。
4) 判断这个状态序列的第一个元素对应的交换是否在禁忌表里，如果不在，跳转到 7。否则跳转到 5。
5) 进行特赦：将这个状态序列的第一个元素对应的交换从禁忌队列中取出，加入禁忌队列头部。如果队列此时长度溢出，则弹出队列尾部元素。之后跳转到 8。
6) 遍历这个状态序列，找到第一个元素使得其对应的交换不在禁忌队列中。之后跳转到 7。
7) 更新禁忌队列：将找到的这个元素对应的交换加入禁忌队列头部。如果队列此时长度溢出，

则弹出队列尾部元素。之后跳转到 8。

8) 如果当前迭代总次数达到了最大迭代次数T，则算法终止。否则跳转到 2。

在 burma14 数据中，我设置最大迭代次数$T = 500$，选取状态序列长度$K = 20$，禁忌队列最大长度$M = 5$，成功达到了理论最优值 3323，运算时间为 0.27s。运行结果随迭代次数的变化如图 1 所示。

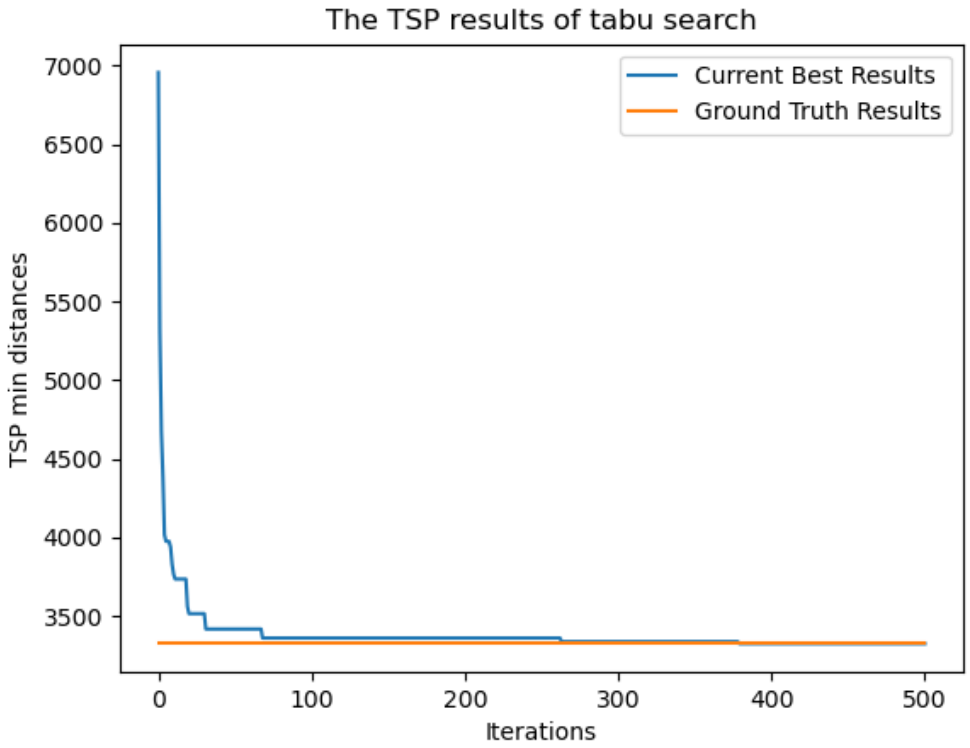

图 1　禁忌搜索算法的运行结果随迭代次数的变化示意图

可以看出，在迭代的初期，算法得到的最优解会迅速改进。在迭代一段时间之后，算法得到的最优解会暂时保持不变，然后在某次迭代突然下降，图像呈阶梯状。这也符合禁忌搜索算法的特点：先大幅度持续改进结果，然后陷入局部最优解，再通过随机化和禁忌、特赦机制跳出局部最优解。

2.2　参数对算法效果、速度的影响

我采用 att48 数据进行实验，其节点个数$N = 48$，迭代次数$T = 10000$，其理论最优解为 10628。首先，我固定禁忌队列长度 M=10，比较不同状态序列长度 K 对算法造成的影响，结果见表 1。

状态序列长度K	10	20	50	100	200	500
结果	16576	13450	11523	**10972**	11103	12390
耗时	6.02s	9.11s	17.35s	32.47s	61.95s	142.75s

表 1 不同状态序列长度 K 的运行结果对比

可以看出，状态序列长度适中才能让算法效果最好：状态序列长度过小的话，由于每次搜索的范围有限，算法不容易跳出局部最优解，不容易得到好的效果。当状态序列长度过大的时候，也并不会对跳出局部最优解有很大的帮助，还会大大增加时间开销，因为求状态序列并排序的复杂度是$O(KN + KlogK)$。

之后，我固定状态序列长度$K = 100$，比较不同禁忌队列长度M对算法造成的影响，结果见表 2。

禁忌队列长度M	3	5	10	20	50	100
结果	11163	11173	**10972**	11062	11509	12100
耗时	32.45s	32.58s	32.47s	32.58s	32.87s	32.51s

表 2 不同禁忌队列长度 M 的运行结果对比

可以看出，禁忌队列长度适中才能让算法效果最好。当禁忌队列长度过小的时候，被禁忌的搜索项很容易就会离开禁忌队列，让算法难以吸取之前搜索的教训，造成效果不佳。而禁忌队列长度过大的时候，被禁忌的搜索项很难离开禁忌队列，导致难以跳出局部最优解。而由于算法的主要时间开销来自求状态序列和排序，因此不同的禁忌队列长度对算法效率影响不大。

2.3 结果汇总

在 burma14、att48、eil101、a280 四组数据下，我调整了算法的参数，得到的最优结果和对应参数见表 3。

参数\数据集	burma14	att48	eil101	a280
理论最优解	3323	10628	629.00	2579.00
算法得到的最好解	3323	10972	722.00	3874.64
性能比	1.00	1.03	1.15	1.50
耗时	0.27s	32.47s	272.20s	2561.62s
迭代次数 T	500	10000	10000	10000
状态序列长度 K	20	100	500	2000
禁忌队列长度 M	5	10	5	5

表 3 禁忌搜索算法求解 TSP 问题的结果汇总

可以看出，较好的状态序列长度随着问题规模的扩大在不断增加，而较好的禁忌队列长度则没有明显变化，都是 5 到 10 之间。随着问题规模的扩大，算法的效果不断变差，由于问题越来越复杂，跳出局部最优解也更难。而算法的耗时在不断增加，因为算法每次迭代都需要$O(KN + KlogK)$的时间来求解状态序列和排序，并且随着问题规模 N 的不断变大，较好的状态序列长度也在不断变大。

3 遗传算法

3.1 算法介绍和实现方法

遗传算法（Genetic Algorithm）是 Holland[3]在 1970 年代提出的算法。此算法借用了生物学中种群交配、基因变异、自然选择和进化的内容，通过交叉、变异和选择来不断改进种群中的实例，进而收敛到全局最优解。我的实现方法如下：

1) 初始化阶段，设 TSP 问题的节点个数为N，初始种群大小为M，我随机初始化M个城市访问序列作为初始种群，求出其总距离，并且记录里面距离最短的访问序列作为初始最优解。
2) 进入遗传算法主循环，总共循环T次。为了保证遗传算法收敛到全局最优解，我们对几个步骤的执行顺序做了调整：首先在种群中进行交叉、然后进行变异、然后对种群的所有访问序列按照总距离从小到大排序更新最优解、最后求适应度函数进行种群选择。
3) 交叉阶段：我将 1 到M这M个访问序列随机分成$\frac{M}{2}$组，每组 2 个序列。对于每组数据，按照交叉概率p_c进行交叉。我的交叉方法是两点交叉法，也就是选择两个端点，对组中两个序列在这两个端点之中的子序列进行交叉。交叉之后两个序列在端点两边的部分可能会产生重复，因此还需要进行去重处理，即记录下重复的位置，使交叉双方重复的节点进行交换。
4) 变异阶段：对于种群中的每个访问序列，我们都按照变异概率p_m进行变异。我的变异方法是选择两个端点，对两个端点之中的子序列进行取反。
5) 更新最优解阶段：对于现在的种群，我们将其按照总距离由小到大排序，并且更新最优解和最优值。
6) 选择阶段：首先，需要设定适应度函数。我设计了两种不同的适应度函数：一种是距离适应度函数，即总距离倒数的平方；一种是排序适应度函数，即将种群按照总距离由大到小排序，总距离最大的适应度为 1，第二大的适应度为 2，以此类推。之后，我们采用轮盘赌算法进行选择，保留M个序列。

在 burma14 数据中，我设置最大迭代次数$T = 200$，初始种群大小$M = 50$，交叉概率$p_c = 1$，变异概率$p_m = 0.05$，适应度函数为距离适应度函数，成功达到了理论最优值 3323，运算时间为 0.40s。运行结果随迭代次数的变化如图 2 所示。

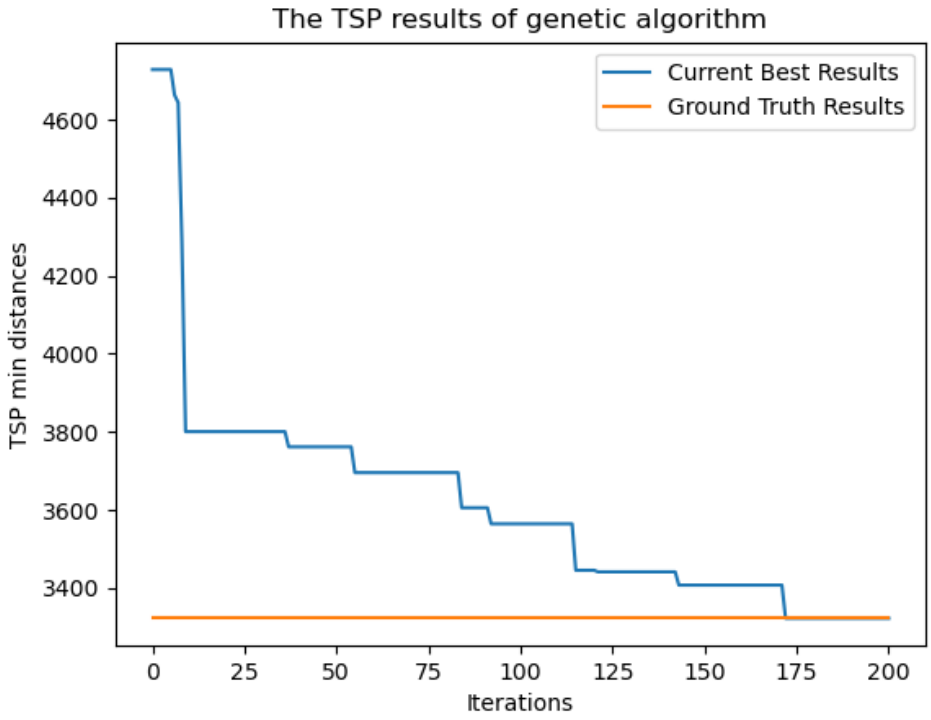

图 2 遗传算法的运行结果随迭代次数的变化示意图

可以看出，在迭代的初期，算法得到的最优解会迅速改进。在迭代一段时间之后，算法得到的最优解会暂时保持不变，然后在某次迭代突然下降，图像呈阶梯状。这也符合遗传算法的特点：首先迅速优化种群，然后通过随机化的交叉、变异机制寻求跳出局部最优解。

3.2 参数对算法效果、速度的影响

我采用 att48 数据进行实验，其节点个数$N = 48$，迭代次数$T = 10000$，其理论最优解为 10628。首先，我分析初始种群大小M对结果的影响，固定交叉概率$p_c = 1$, 变异概率$p_m = 0.05$， 适应度函数为距离适应度函数。结果见表 4。

初始种群大小M	10	20	50	100	200	500
结果	12807	10897	11045	**10771**	11156	18397
耗时	10.04s	14.15s	32.10s	92.02s	178.53s	549.03s

表 4 不同初始种群大小度 M 的运行结果对比

可以看出，初始种群大小适中才能让效果最好。初始种群太小的话，可供选择、交叉和变异的个体有限，交叉和变异的范围也不够大，搜索不充分，难以跳出局部最优解。初始种群过大的话，会让交叉、变异过于频繁，干扰整体的搜索方向，而且运算速度也大大减缓。

其次，我分析交叉和变异机制对结果的影响。我固定初始种群大小 M=100，适应度函数为距离适应度函数，变异概率$p_m = 0.2$。不同交叉概率p_c对应的结果见表 5。

交叉概率p_c	0	0.5	0.7	1
结果	35787	13207	11082	**10771**
耗时	20.02s	56.47s	66.73s	92.02s

表 5 不同交叉概率p_c的运行结果对比

我固定初始种群大小$M = 100$，适应度函数为距离适应度函数，交叉概率$p_c = 1$。不同变异概率p_m对应的结果见表 6。

变异概率p_m	0	0.05	0.1	0.2
结果	21804	11154	10905	**10771**
耗时	70.37s	77.50s	82.86s	92.02s

表 6 不同变异概率p_m的运行结果对比

可以看出，交叉和变异机制都必不可少，没了任何一个机制，整体的效果都会大打折扣。

3.3 结果汇总

在 burma14、att48、eil101、a280 四组数据下，我调整了算法的参数，得到的最优结果和对应参数见表 7。

参数\数据集	burma14	att48	eil101	a280
理论最优解	3323	10628	629.00	2579.00
算法得到的最好解	3323	10771	677.09	2987.29
性能比	1.00	1.01	1.08	1.16
耗时	0.40s	92.02s	137.73s	1574.82s
迭代次数T	200	10000	10000	10000
初始种群大小M	50	100	200	500
交叉概率p_c	1	1	0.5	1
变异概率p_m	0.05	0.2	0.05	0.05
适应度函数	距离	距离	距离	距离

表 7 遗传算法求解 TSP 问题的结果汇总

可以看出，当问题规模不断增大的时候，要得到较好的结果所需的初始种群大小也要不断增大，因为当问题越来越复杂的时候，增加每轮迭代的初始种群大小能有效保证种群个体基因的多样性，便于找到最优解。而交叉概率和变异概率则看不出特定的规律，交叉概率在$[0.5, 1]$之间，变异概率在 0.2 以下，都是较为合理的选择，需要去不断尝试以得到最优参数。

3.4 适应度函数对算法效果、速度的影响

最终，我分析适应度函数对结果的影响。对于 att48，eil101，a280 三组不同的数据，我选取使得其效果最好的初始种群大小M，交叉概率 pc 和变异概率 pm，对比两种适应度函数的结果。结果见表 8。

适应度函数/数据集		att48	eil101	a280
距离	结果	10771	677.09	2987.29
	耗时	92.02s	137.73s	1574.82s
排序	结果	10771	677.09	2987.29
	耗时	88.86s	137.68s	1572.13s

表 8 不同适应度函数的运行结果对比

可以看出，当种群大小、交叉概率、变异概率都选取的较为合适的时候，两个适应度函数差别并不大，两者得到的最优解相同，而排序适应度函数耗时略少。

4 模拟退火算法

4.1 算法介绍和实现方法

模拟退火算法（Simulated Annealing）是 Metropolis 等[4]在 1953 年提出的算法。1983 年，Kirkpatrick 等[5]将其扩展到组合优化领域。这一算法模拟金属冶炼的固体退火原理，在温度较高时，固体内部粒子趋于无序，而随着温度降低，固体内部粒子愈发有序。这一算法设定初始温度较高，且温度会指数下降。当温度较高时，算法会倾向于不断探索新的结果，而当温度较低时，算法倾向于维持现状，除

非遇到更好的结果。我的实现方法如下：

1) 初始化阶段，设节点个数为N，随机生成一个访问序列作为初始解和初始最优解。设定初始温度$T = T_{start}$。
2) 外循环阶段，更新T使得$T = kT$。其中k为衰减系数，是一个$(0,1)$之间的常数。当T小于终止温度T_{end}时，算法运行结束。
3) 内循环阶段，循环M次，每次使用 2-opt 方法在当前解x_t的邻域中随机找一个新解x_{t+1}。若新解比当前解更优，则用新解替换当前解。否则，令当前解x_t的总距离为d_t，新解x_{t+1}的总距离为d_{t+1}，算法有$e^{\frac{-(d_{t+1}-d_t)}{T}}$的概率用新解替换当前解。

在 burma14 数据中，我设置初始温度$T_{start} = 100$，终止温度$T_{end} = 0.01$，内循环次数$M = 50$，衰减速率$k = 0.99$，成功达到了数据集报告的最优值 3323，运算时间为 25.71s。迭代结果变化如图 3 所示。

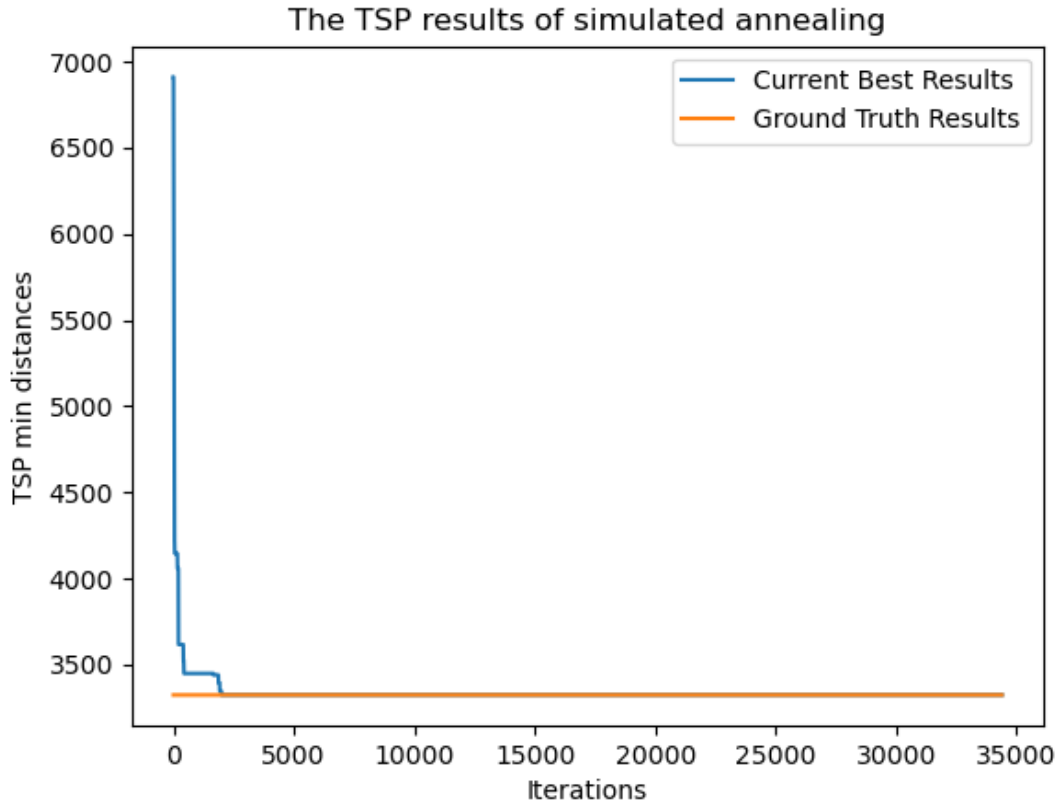

图 3 模拟退火算法的运行结果随迭代次数的变化示意图

可以看出，迭代结果的改进主要源自初期，也就是温度T较大的时候。这个时候比较容易跳出局部最优解。而整体图像呈阶梯状：算法利用高温震荡机制跳出局部最优解，然后利用贪心搜索机制快速改进结果。

4.2 参数对算法效果、速度的影响

当温度$T = 0.1$时，如果$d_{t+1} - d_t = 1$，则此时替换当前解的概率为$4.54 * 10^{-5}$，可以忽略不计，因此我取终止温度$T_{end} = 0.1$。而其余的参数，包括初始温度T_{start}，衰减系数k，内循环次数M，都需要进行实验来确定。我使用 att48 数据来探讨不同参数对算法结果的影响。

首先，我固定初始温度$T_{start} = 500$，衰减系数$k = 0.99$，分析不同内循环次数M对算法结果的影响。结果见表 9。

内循环次数 M	20	50	100	200	500	1000	2000
结果	27646	27646	14678	**11630**	15046	15514	21074
耗时	0.84s	2.02s	4.09s	8.31s	20.44s	41.11s	83.47s

表 9 不同内循环次数M的运行结果对比

可以看出，当内循环次数M适中的时候，才能得到比较好的结果。当M过小的时候，搜索的范围不够大，不容易跳出局部最优解。当M过大的时候，过度的震荡可能会干扰整体的搜索方向，而且搜索耗时过大。

之后，我固定内循环次数$M = 200$， 衰减系数$k = 0.99$，分析不同初始温度T_{start}对算法结果的影响。结果见表 10。

初始温度T_{start}	10	20	50	100	200	500	1000
结果	15294	14954	14273	16087	12888	**11630**	26339
耗时	4.46s	5.10s	5.95s	6.57s	7.28s	8.31s	8.67s

表 10 不同初始温度T_{start}的运行结果对比

可以看出，当初始温度T_{start}适中的时候，才能得到比较好的结果。当初始温度T_{start}过小的时候，进行非贪心选择的概率会比较小，导致不容易跳出局部最优解。当初始温度T_{start}过大的时候，过度的震荡可能会干扰整体的搜索方向，也会导致搜索耗时过大。

4.3 结果汇总

在 burma14、att48、eil101、a280 四组数据下，我调整了算法的参数，得到的最优结果和对应参数见表 11。

参数\数据集	burma14	att48	eil101	a280
理论最优解	3323	10628	629.00	2579.00
算法得到的最好解	3323	11630	843.76	5448.17
性能比	1.00	1.09	1.34	2.11
耗时	4.37s	8.31s	76.50s	117.68s
内循环次数M	50	200	500	2000
初始温度T_{start}	100	500	5	5
终止温度T_{end}	0.1	0.1	0.1	0.1
衰减系数k	0.99	0.99	0.98	0.99

表 11 模拟退火算法求解 TSP 问题的结果汇总

可以看出，衰减系数k选择 0.98 和 0.99 这种较大的系数较好，否则会导致温度下降过快，不能充分搜索，影响搜索效率。内循环次数M和初始温度T_{start}的选择则依赖问题本身的性质，如果局部最优解难以跳出，则最好选择较大的内循环次数和初始温度以扩大搜索范围，否则最好选择较小的内循环次数和初始温度以防止过度震荡影响算法效果。

5 分析与总结

5.1 不同算法的效果、速度对比

我在表 12 中展示了对于 burma14、att48、eil101、a280 这四组数据，禁忌搜索算法、遗传算法和模拟退火算法三个算法得到的最好结果和耗时。

数据集		burma14	att48	eil101	a280
理论最优解		3323	10628	629.00	2579.00
禁忌搜索	算法得到的最好解	3323	10972	722.00	3874.64
	性能比	1.00	1.03	1.15	1.50

	耗时	0.27s	32.47s	272.20s	2561.62s
遗传算法	算法得到的最好解	3323	10771	677.09	2987.29
	性能比	1.00	1.01	1.08	1.16
	耗时	0.40s	92.02s	137.73s	1574.82s
模拟退火	算法得到的最好解	3323	11630	843.76	5448.17
	性能比	1.00	1.09	1.34	2.11
	耗时	4.37s	8.31s	76.50s	117.68s

表 12 禁忌搜索算法、遗传算法和模拟退火算法求解 TSP 问题的结果对比

可以看出，首先，随着问题规模的增大，几种算法求解的效率和效果都不断变差，原因是问题更加复杂，更难得到最优解，同时也需要更多的搜索和尝试来跳出局部最优解，耗时飞速上涨。其次，对比三种算法，模拟退火算法耗时最少，同时效果也最差；遗传算法耗时最多，同时效果也最好；禁忌搜索算法效果位于两者之间，耗时也较大。由于遗传算法引入了遗传-变异机制，能够大大扩大搜索的范围，更容易跳出局部最优解，效果最好，但是因为这两个操作对于每条访问路径，都需要$O(N)$的时间完成，因此耗时最高。禁忌搜索和模拟退火都是在 2-opt 邻域上进行搜索，速度较快，但是搜索范围就受限了。不过禁忌搜索可以在一次迭代中搜索若干个候选解进行优中选优，效果更好。而模拟退火则一次只能搜索一个解，搜索范围受限，因此效果不佳。

5.2 总结

禁忌搜索、遗传算法和模拟退火算法都是组合优化的经典算法，三者都充分利用了随机性以跳出局部最优解，以期解决组合优化问题。这三种算法都有一些较为重要的参数，它们的选择可以大大影响算法效率和效果，我在此将其分为两类：一类是长度类参数，一类是其他参数。前者包括禁忌搜索的状态序列长度K、禁忌队列长度M、遗传算法的初始种群大小M，模拟退火算法的内循环次数M、初始温度T。这类参数的选取往往是适中最佳：当选取参数太小的时候，会导致搜索不够充分，难以跳出局部最优；当选取参数太大的时候，会导致搜索范围过大，效率下降。后者包括遗传算法的交叉概率p_c和变异概率p_m，还有模拟退火算法的衰减系数k。这类参数的选取则是在一个合理范围内均可，不过其选取和问题本身关系较大，没有特别强的规律，需要做充分的尝试才能找到最好的参数。

对比三个算法，可以看出，最为复杂的遗传算法的效果最好，但是速度最慢。复杂度适中的禁忌搜索效果次之，而速度也较慢。而最为简洁的模拟退火算法速度最快，效果也最差。要想改进算法的效果，往往需要引入更复杂的机制以扩大搜索范围，以期跳出局部最优解，这样就会导致算法速度变慢。在解决实际问题时，需要根据问题的特性和实际需求，权衡好效果和效率，选择合适的算法。

参考文献

[1] Reinelt, G. (1995) TSPLIB. https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[2] Glover, F. (1986) “Future Paths for Integer Programming and Links to Artificial Intelligence,” Computers and Operations Research, Vol. 13, pp. 533-549.

[3] Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[4] Steinbrunn M ,Moerkotte G, Kemper A. Heuristic and Randomized Optimization for the Join Ordering Problem[J] . The VLDB Journal , 1997 , 6 (3) :8 - 17.

[5] Kirkpatrick S, Gelatt CD Jr, Vecchi MP. Optimization by simulated annealing. Science. 1983 May 13;220(4598):671-80. doi: 10.1126/science.220.4598.671. PMID: 17813860.