人工智能实验报告
局部搜索与模拟退火

小组　lgd-cs
赖启东*　14347058
黎丁嘉　15336077
李振宇　15336092
刘键涵　15336110
洪培衔　15336059
2015级计算机科学与技术
School of Data and Computer Science, Sun Yat-Sen University, China
laiqd@mail2.sysu.edu.cn

December 26, 2017

Abstract

本次实验我们组采用局部算法(Local Search, LS)和模拟退火算法(Simulate Anneal, SA)解决旅行商问题(Travel Saleman Problem, TSP)。我们采用了三个常见的邻域算子，采用随机部分邻域搜索策略，拿数据eil101.tsp进行测试。局部搜索得出结果与最优解差距在5%左右，模拟退火得出解与最优解的误差在3%内。我们发现这两种算法都可以有效得出较好解，其中模拟退火更好一些。

1 Introduction

1.1 Problem Description

给定一个无向图$G = (N, E)$, 其中$N = \{1, 2, \ldots, n\}$是顶点集$E = \{(i, j) : i, j \in N, i \neq j\}$是边集，每条边$(i, j)$有一个正数表示旅行花费$c[i, j]$。路径$r = (v_1, v_2, \ldots, v_n)$是一个$N$中每个点各出现一次的序列，其中$v_i (i \in [1, n])$表示序列中第$i$项。序列中第一项$v_1$是起点，旅行商按照路径顺序访问每个点后回到起点，(最后从v_n回到v_1)。路径r的花费$cost_r = c[v_n, v_1] + \sum_{i=1}^{n-1} c[v_i, v_{i+1}]$。TSP就是给定这样一个无向图，求一条花费最小的路径。

1.2 Algorithm Review

1.2.1 Local Search

局部搜索是一种解决最优化问题的启发式算法[1]。局部搜索从一个初始解开始，通过邻域算子操作，产生邻居解。然后判断邻居解的质量，再根据某种策略来选择邻居作为当前解。再拿当前解进行邻域操作。如此重复以上过程直到达到终止条件。

1.2.2 Simulate Anneal

模拟退火属于高级一些的局部搜索的一种，**本质上仍是局部搜索**。所谓退火是冶金专家为了达到某种特种晶体结构重复将金属加热或冷却的过程，过程中控制了温度T。模拟退火算法基本思想是在系统朝着能量减少的趋势这样一个变化过程中，偶尔允许系统跳到能量较高的状态，以避开局部最小点，最终稳定在全局最小点。

算法流程如下：

(1)随机挑选一单元k，并给它一个随机的位移，求出系统因此而产生的能量变化ΔE_k

(2)若$\Delta E_k \leq 0$，该位移可采纳，变化后的系统状态作为下次变化的起点;若$\Delta E_k > 0$，位移后的状态可采纳概率

$$P_k = \frac{1}{1 + e^{-\frac{\Delta E_k}{T}}} \tag{1}$$

式中T为温度，然后从$(0,1)$区间均匀分布的随机数中挑选一个数R。若$R < P_k$，则将变化后的状态作为下次的起点;否则，将变化前的状态作为下次起点。

(3)转第(1)步继续执行，直到平衡状态为止。

2 Algorithm Design

2.1 Initial Solution

初始解为$r = (1, 2, \ldots, n)$。初始解的好坏对最后结果影响不大，如此构造初始解较简单。

2.2 Local Search

其实LS算法就是不停迭代直到达到收敛条件。我们设置的收敛条件是连续$baditeration$次迭代没有更新最优解。

因为邻域比较大，算子操作一次的时间复杂度较大，花费时间长，所以只在邻域搜寻部分解。采取随机搜索的方式，令$SearchRatio$为搜索的邻域占总邻域大小的比例，$SearchAmount$为搜索邻域大小。

算法图中5是计算往旁边邻域搜索的解的个数，8到14是具体搜索的过程，s^*是搜索中的一个邻域解，ss为邻域解中最好的一个。21是判断是否达到收敛条件。

Algorithm 1 LS

$s \leftarrow$初始解
$ans \leftarrow s$
$iteration \leftarrow 0$
repeat
  $SearchAmount \leftarrow SearchRatio*$s的邻域大小
  $j \leftarrow 0$
  $ss \leftarrow$空解
  repeat
    在s邻域内随机选择一个解s^*
    if s^*比ss好 **then**
      $ss \leftarrow s^*$
    end if
    $j \leftarrow j + 1$
  until $j > SearchAmount$
  if ss比s好 **then**
    $iteration \leftarrow 0$
  else
    $iteration \leftarrow iteration + 1$
  end if
  $s \leftarrow ss$
until $iteration > baditeration$
return ans

2.3 Simulate Anneal

模拟退火的过程大概如下面算法框架所示。我们设置的算法结束条件是温度从初温T_0**降到0**或者是迭代次数达到$iteration_{max}$。因为初温的设计与初始解有关，有可能初始解较大，降温降到一定程度就已经收敛了，没必要继续下去，所以设置了最大迭代次数$iteration_{max}$。

传统模拟退火框架为两个循环，分为外循环和内循环。外循环从5到31，是控制迭代与温度的过程。内循环是从7到28。18是计算能量变化量，19和22是两种不同情况对邻域解的接收策略。对比局部搜索发现，不同是的22这里以一定概率接受劣解。28这里是降温。

Algorithm 2 SA

$s \leftarrow$初始解
$ans \leftarrow s$
$T \leftarrow T_0$
$iteration \leftarrow 0$
repeat
  $i \leftarrow 1$
  repeat
    $SearchAmount \leftarrow SearchRatio*$s的邻域大小
    $j \leftarrow 0$
    $ss \leftarrow$空解
    repeat
      在s邻域内随机选择一个解s^*
      if s^*比ss好 **then**
        $ss \leftarrow s^*$
      end if
      $j \leftarrow j + 1$
    until $j > SearchAmount$
    $dE \leftarrow ss$的花费$-s$的花费
    if $dE < 0$ **then**
      $s \leftarrow ss$
      更新ans
    else
      if $e^{\frac{-dE}{T}} \leq$ 一个$[0,1]$的随机实数 **then**
        $s \leftarrow ss$
      end if
    end if
    $i \leftarrow i + 1$
  until $i > inneri$
  $T \leftarrow T * \alpha$
  $iteration \leftarrow iteration + 1$
until $T \leq 0$或$iteration > iteration_{max}$
return ans

2.4 Neighborhood Operators

2.4.1 Swap

交换路径中两个点。例如$r = (5, 4, 3, 2, 1)$，交换2与5后变为$r = (2, 4, 3, 5, 1)$。邻域大小为$O(n^2)$，一次操作时间复杂度为$O(1)$，因为交换两个点，路径花费改变量仅为这两点各与其相邻点所连的边的改变，一共四条边，所以做一次操作时间是常数级别的。

2.4.2 Relocation

将路径中一个点移动到路径中的某个位置。例如$r = (5, 4, 3, 2, 1)$，把3移动到1后面变为$r = (5, 4, 2, 1, 3)$。邻域大小为$O(n^2)$，一次操作时间的复杂度为$O(n)$。

2.4.3 2-opt

将路径中一段连续的子序列翻转。先选择两个位置w_1和$w_2(w_1 \leq w_2)$，然后把$r_{w_1}, r_{w_1+1}, \ldots, r_{w_2}$这段翻转。例如$r = (5, 4, 3, 2, 1)$，选择$w_1 = 1$，$w_2 = 3$，则翻转后为$r = (3, 4, 5, 2, 1)$。邻域大小为$O(n^2)$，一次操作时间复杂度为$O(n)$。

3 Computational Experiments

3.1 Parameter Settings

3.1.1 Local Search

设当前已经有连续$iteration$次迭代不更新了，那么搜索邻域比例$SearchRatio = iteration * 5\%$。因为要尽力找更优解，当没法更新的话就要扩大邻域搜索范围，提高邻域搜索的比例，才有更大可能性搜到更优解。所以把比例设置成连续不更新迭代次数的线性函数。而设置上限5%是因为邻域大小是$O(n^2)$,一个操作是$O(n)$，一次全邻域探索则是$O(n^3)$，很耗时，所以只取了比较小的比例。

最大连续不新迭代次数$baditeration = 50000$。

3.1.2 Simulate Anneal

初温T_0设置为初始解的花费。搜索邻域比例$SearchRatio = 5\%$。降温系数$\alpha = 0.98$。内循环迭代次数$inneri = 1000$，外循环最大迭代次数$iteration_{max} = 50$。

3.2 Datasets

选取的数据为eil101.tsp。这个数据里有101个点，符合老师点数大于100的要求。

3.3 Environment

我们的程序使用C++语言编写，实验运行在Intel Core i7-4510U 2.0GHz的CPU，8GB RAM，Ubuntu 14.04LTS操作系统的个人电脑上,使用了g++O4优化。

3.4 Results

如图3.4为局部搜索运行结果，图3.4为模拟退火运行结果。两幅图中左边为选取算法运行的效果，右边为最优解的情况。这个程序**边迭代边描绘**当前最优解的路线。运行程序可以看到随着迭代的进行，路径交叉越来越少。这两幅图是最后的结果，其实都只有一个交叉点了。对比最优解的图可以看出大致轮廓基本一致，基本上是交叉处的路线与最优解不同，说明算法求出来的路线相当让人满意。

再来看**量化**的数据。局部搜索与最优解的误差约为5%，而模拟退火与最优解差距不到3%，在这个结果上表现优异。

同时，UI中加入重要参数的接口以及不同测试数据的输入端。这样方便测试者根据不同数据进行调参之类，适应性强。代码中使用C++内置容器类型，可以适应不同程序规模，有很强**鲁棒性**。

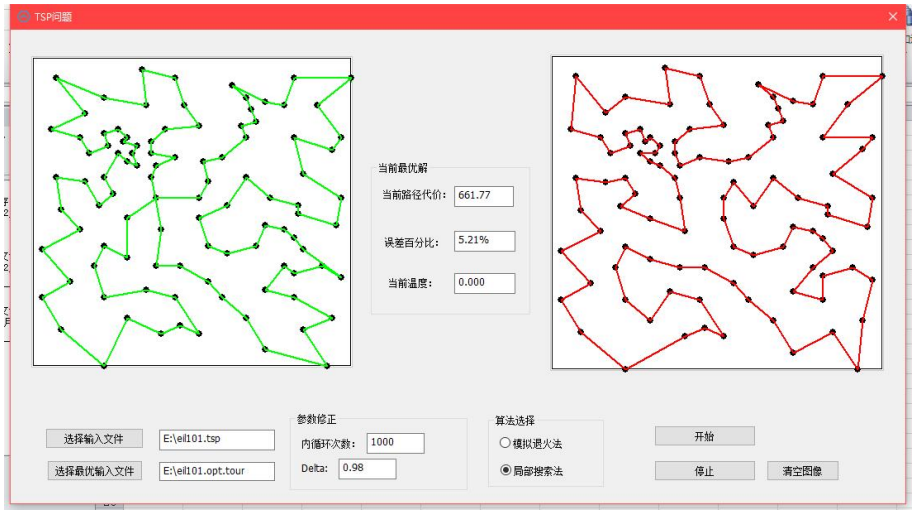

Figure 1: 图1

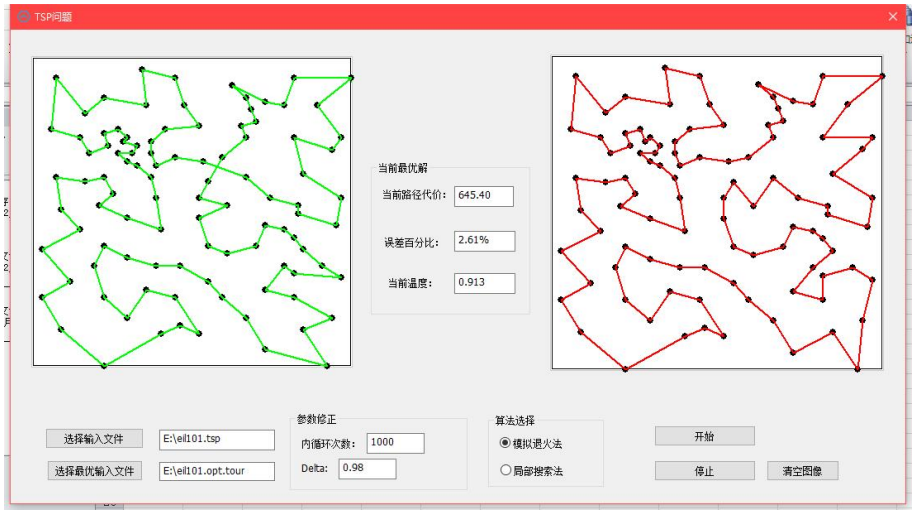

Figure 2: 图2

表1是进行10次局部搜索和模拟测试的结果。我们求出了平均值与方差，可以看到两种算法方差都不大，模拟退火的非常小，可以看出模拟退火更具稳定。

官方提供的最优解是629。可以计算出两种算法得出最好解与官方解的差距：

$$Error_{LS} = \frac{661.77 - 629}{629} = 5.21\%$$

$$Error_{SA} = \frac{645.4 - 629}{629} = 2.61\%$$

老师要求不超过10%，即$629 * (1 + 10\%) = 691.9$。从表中可以看出结果均小于这个值，说明稳定满足老师要求。

从表中可以看出模拟退火得出解的质量全面优于局部搜索，但是模拟退火所用的时间是大于局部搜索的。其实各有优劣。但我们更关注解的质量，所以认为模拟退火更好一点。

Table 1: 表1

测试次数	局部搜索		模拟退火	
	花费	时间	花费	时间
1	680.64	10.86	649.47	17.56
2	**661.77**	10.84	**645.4**	18.1
3	680.64	10.88	648.64	17.85
4	**661.77**	10.88	647.17	18.13
5	**661.77**	10.85	**645.4**	18.05
6	**661.77**	10.83	645.46	17.69
7	661.93	10.87	647.17	17.95
8	**661.77**	10.79	**645.4**	17.88
9	**661.77**	10.94	647.05	18.05
10	661.93	10.93	647.17	18.02
平均值	665.576	10.867	646.833	17.928
方差	56.73486	0.001801	1.865241	0.030956
最优值	661.77		645.4	

4 Conclusion

通过这次实验，我们学到了局部搜索与模拟退火的基本框架，写了三个常用算子，尝试调参。我们初步了解智能搜索的思路，应用到经典的TSP问题。我们发现模拟退火比局部搜索好一点。

我们初步使用智能搜索算法发现这看似不智能的算法实际上能得出很好的解，比预期强很多。希望在后面的学习中能学到更多好算法提升解的质量。

5 实验分工

赖启东负责SA和LS算法核心部分。
李振宇将算法移植到MFC上并完成初版程序。
黎丁嘉优化界面，dubeg，调整SA算法参数完成最终版程序。
洪培衔想出新算子添加入SA，加快速度。
刘键涵调试程序，调整算法参数，提高最优解。
实验报告由赖启东与黎丁嘉撰写，平时实验过程记录报告由黎丁嘉和李振宇完成。

References

[1] 朱福喜, "人工智能基础教程(第二版)," 2011.