2N皇后说明文档

PB14011009 邹易澄

操作说明

本目录下有2个cpp文件、2个Windows系统的exe可执行文件、2个linux系统的可执行文件、本说明文档。

Windows系统：将input.txt文件放在此目录下，双击exe可执行文件，会在本目录下生成结果文件。

Linux系统：将input.txt文件放在此目录下，在终端下执行，会在本目录下生成结果文件。

程序说明

本实验我采用了爬山算法和模拟退火算法。

为了优化空间复杂度，原本需要一个n*n的二维数组记录棋盘的状态，现在只用一个长度为n的数组，第i个元素表示第i行皇后的位置(列号)，复杂度为O(n)。这样还有一个好处，如果保证这n个元素的值两两不同，即列号不同，则冲突值的计算只要考虑对角线，大大减少了搜索空间。维护这个性质也很方便，状态更新只需要交换两行的数值，就能保证每一列的元素均不相同。

为了优化时间复杂度，我在程序一开始生成一部分没有冲突的皇后，这些皇后的位置将被固定，不能被交换位置。这样，可以大大减少解空间，使之能够快速收敛。

计算冲突值的函数原本需要O(n^2)，每次传入一个棋盘状态，得到冲突的皇后数目，算法如下：

```
long h(const vector<long> &status)
{
    long num = 0;
    for (long i = 0; i < status.size(); ++i)
        for (long j = i + 1; j < status.size(); ++j)
            if (abs(status[i] - status[j]) == j - i)
                num++;
    return num;
}
```

我们可以将这个过程降为O(1)，代价是要维护两个`2*n-1`的数组，记录两个方向的所有对角线上的皇后数目，还需要记录原先状态的冲突数，不过空间复杂度仍然是O(n)没有变。计算新的冲突数时，只需要加上两皇后交换前后的冲突数差值即可。

2N皇后问题只是N皇后问题的一个变换。对于N等于偶数的情况，只需要将N皇后的解关于中轴线做一次对称变换，就能得到一组新的解。对于N等于奇数的情况，只需要生成一组解后，记录皇后位置，再进行一次求解过程，生成与原来的解不冲突的解即可。

- 随机重启爬山算法

爬山算法是一种贪心策略，算法不维护搜索树，因此当前节点的数据结构只需要记录当前状态和它的目标函数值。每当遇到更优的状态，便转移。如果没有更优的状态，则允许侧向移动一定的次数。若侧向移动次数超过限制，或者没有上山以及平原的路，则采用随机重启策略，重新生成一个初始解，重新上山。

由于爬山法本身只记录当前状态和它的目标函数值，因此空间复杂度为O(n).

为了优化时间复杂度，我采用了首选爬山法，即每次遇到更优的状态就更新，而不是找到最优状态。虽然收敛速度没有最陡爬山法快，但是在有上千个后继结点的情况下，是个更好的策略。

理论上随机重启爬山法找到解的概率为1.

- 模拟退火算法

模拟退火算法，与随机重启爬山算法不同，它能在一定概率下转移到较差的状态。随着时间的推移，转移到较差状态的概率逐渐变小。可以证明如果让T下降得足够慢，这个算法找到全局最优解的概率逼近于1。

模拟退火算法与随机重启算法相似，本身只记录当前状态和它的目标函数值，因此空间复杂度为O(n).

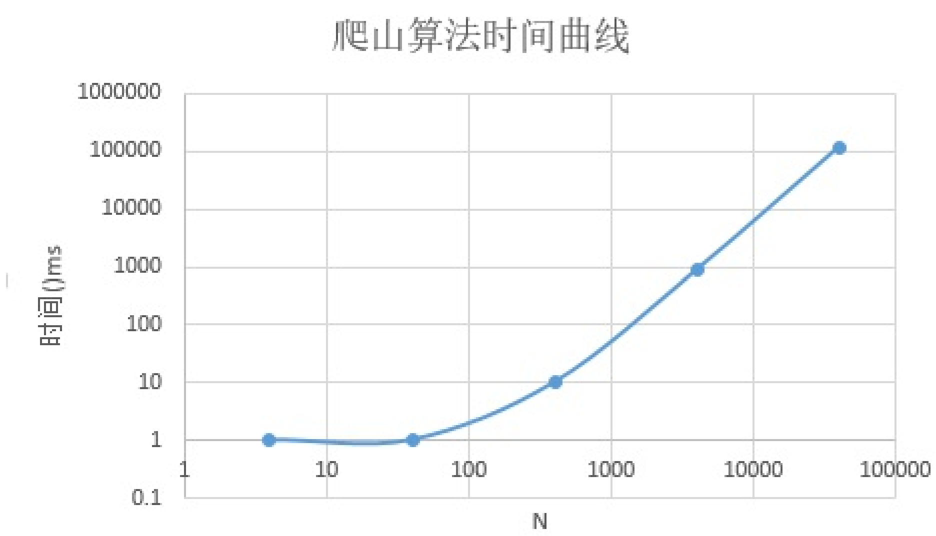
在此次实验中，转移到较差状态的概率为：P(dE) = exp(dE/(T))，其中dE表示冲突值的差值，T表示温度。

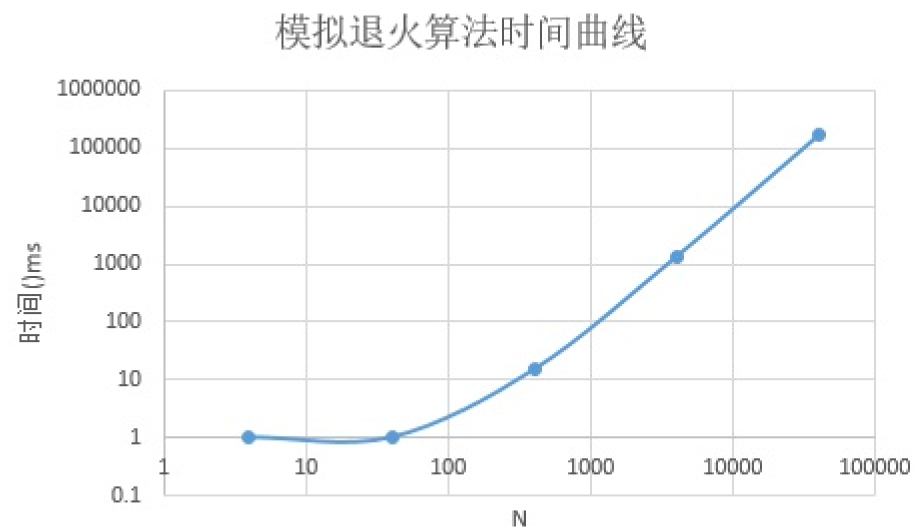
T随着程序运行会逐渐减小，我设计的减小的函数是T = (((N - T) / N) ^ N) * (10 ^ (N - T) / N - 1)，它随着时间的推移，减小的速率越来越慢。

模拟退火算法有可能找不到解，经过大规模的测试，找到解的概率大约为0.86。为了解决这个问题，我加入了随机重启策略。

结果分析

以下是两个算法的时间分析：





在N等于几千时，平均能在1秒内出结果，在N等于几万时，平均能在1分钟左右出结果。可以推得两个算法的时间复杂度均为指数级。

若初始情况设置的固定皇后数目很大时，冲突数很小，只要做微调即可。我设置的数值为N - sqrt(N)，经过这样的优化，百万皇后均能在几秒之内出结果。