人工智能

实验一 N 皇后问题

昂伟 *PB11011058*

算法分析

输入: N
输出: 三组可行解
$size\_t\ **NQueen(int\ N)$ 根据 N 的不同调用三个不同的子函数来解决 N 皇后问题以达到最好的运行效果, 降低时间和空间复杂度.

$0 < N < 15$ 调用 $void\ nqueen1(int\ col, intN)$.

$15 <= N <= 3000$ 调用 $void\ nqueen2(size\_t * queen, int\ N)$.

$3000 < N$ 调用 $void\ nqueen3(size\_t * queen, int\ N)$.

时空复杂度分析

$void\ nqueen1(intcol, intN)$ 是一个递归函数, 时间复杂度递推关系为:

$$T_n = n * T_{n-1}$$

所以,$T_n = O(N!)$. 因为是递归函数, 所以需要维护一个栈, 即空间复杂度为 $S_n = O(N)$.

$void\ nqueen2(size_t\ * queen, int\ N)$ 的算法描述为:

```
begin
    while (解有冲突) {
        (1) 产生一个随机解s
        (2) for all i, j: where 皇后s[i]或皇后s[j]
            有冲突:
            (2.1) if (交换s[i],s[j]能减少冲突)
                    then 交换s[i], s[j]
    }
end
```

分析　算法步骤 (1) 产生随机解的时间复杂度为 $O(N)$，步骤 (2) 为一个双层 for 循环, 所以时间复杂度为 $O(N^2)$. 而最外层的 while 循环的最坏循环次数为 $O(N)$. 所以本算法的最坏时间复杂度为 $O(N^3)$. 由于需要维护两个大小为 $2*N-1$ 的数组, 所以空间复杂度为 $S_N = O(N)$.

$void\ nqueen3(size_t\ *queen, int\ N)$ 的时间复杂度为:$T_N = O(n)$. 空间复杂度为 $S_N = O(N)$. 具体分析参见http://dl.acm.org/citation.cfm?id=101343.

总结

当 N 很小时 ($N < 15$), 使用递归算法可以保证相当好的运行效率; 当 N 特别大时 ($N > 3000$), 采用局部搜索算法有非常好的运行效率 (可在数分钟内解决千万皇后问题); 当 $15 < N < 3000$ 时, 采用类爬山算法可以获得较好的运行效率.

总的来说, 本程序可以解决各种规模的 N 皇后问题.