**局部搜索算法的实验分析**

**实验零：补全代码，并将填空内容写在实验报告中**

hill\_climb.hpp

// TODO：状态转移（需要参考state，selection的接口）——1行

state=state.neighbor(permutation[selection.selected\_index()]);

conflicts\_minimize.hpp

// TODO：将变元新值产生的冲突数的评估价值提交给选择器（需要参考selection, problem的接口）——1行

selection.submit(value\_of(-problem.conflicts\_of(selected\_index)));

simulated\_anneal.hpp

/\*\*\*\*\*\*\*\*TODO: 如果新值更高，直接接受，若更低，则以exp(value\_diff / temperature)概率接受\*\*\*\*\*\*\*/

if (value\_diff > 0 || RandomVariables::uniform\_real() < exp(value\_diff / temperature)){

state = new\_state;

}

**实验一：爬山算法求解n皇后问题**

实验目的：掌握爬山算法的思想与实现，探究影响爬山算法效果的因素

实验步骤：认真阅读作业要求说明文档中的说明，调整爬山算法的参数，尝试在5秒钟之内求解皇后数n尽可能大的n皇后问题（仅需输出1个解）

实验结果：

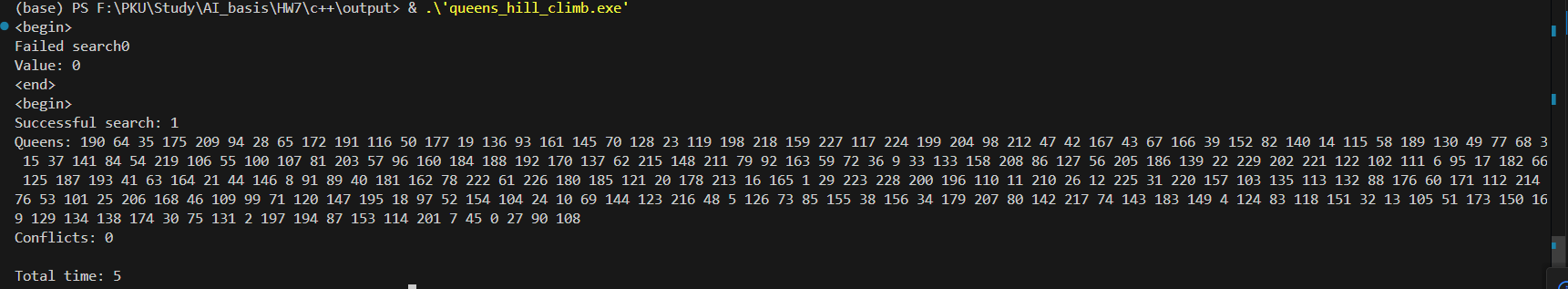
1. 参数设置情况

|  |  |
| --- | --- |
| **参数** | **参数取值** |
| 问题模型 | QueensMoveState |
| 随机重启尝试次数 | 5 |
| 单次最大爬山步数 | 3n/2 |
| 选择算法 | m\_selection |
| 状态估值函数（若选择算法不是轮盘赌则不填此项） | / |

1. 5秒内解出的最多皇后数n为：( 230 )
2. 简述除参数外对算法做出的修改（无修改则填无）：

无

实验结果分析：



经过多次尝试，发现QueensSwapState的平均时间在n为200左右的时候远远长于QueenMoveState;要在n=200左右的问题规模下在5s内解决问题，最多只能失败一次，否则即超时，因此随机重启的次数这一参数不太影响实验结果；单次最大爬山步数如果设置的太大，也会显著影响算法的时间，若是设为4n或以上，完成一次搜索的时间便很长，此时若搜索失败，便会直接超时.且经过多次尝试，发现一次搜索成功的情况比较少，大多数情况下都要进行至少一次重启.因此这里我反其道而行之，将步数设为3n/2，这样可以保证在陷入局部最优的时候算法可以快速抽离重启，从而在严格的时间限制下搜索出解.经过实验，三种选择算法在速度上没有明显差别，选择m\_selection是为了在大规模问题下贪心地选择邻居，使其更快收束，配合上步数的严格限制，可以使算法快速搜索到解.

**实验二：模拟退火算法求解n皇后问题**

实验目的：掌握模拟退火算法的思想与实现，探究影响模拟退火算法效果的因素

实验步骤： 认真阅读作业要求说明文档中的说明，调整模拟退火算法的参数，尝试在5秒钟之内求解皇后数n尽可能大的n皇后问题（仅需输出1个解）

实验结果：

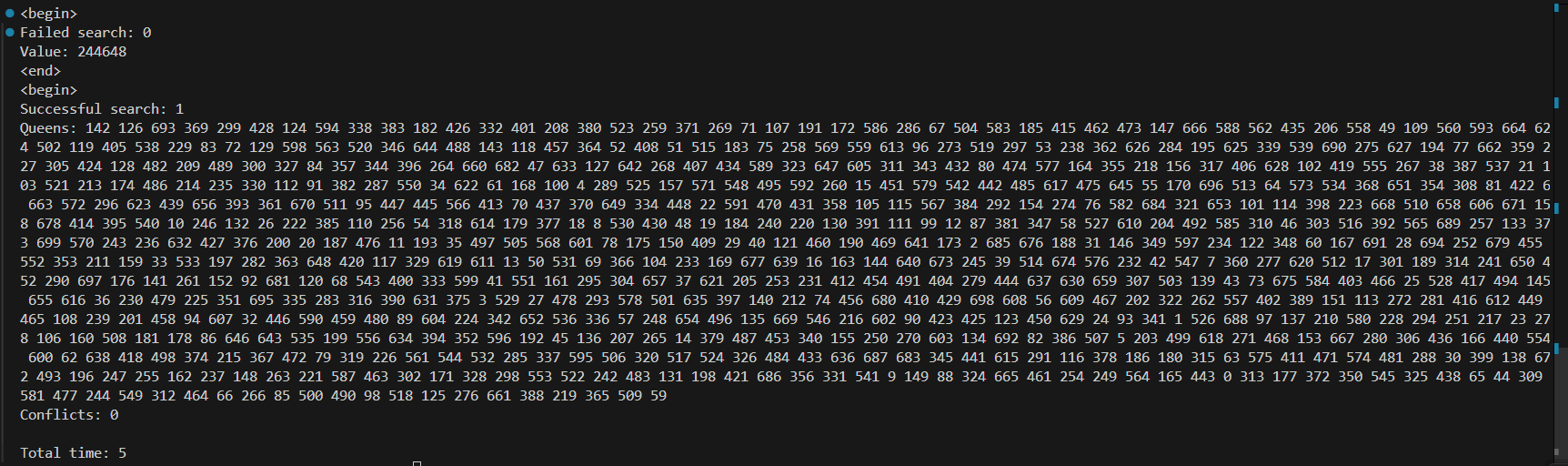
1. 参数设置情况

|  |  |
| --- | --- |
| **参数** | **参数取值** |
| 问题模型 | QueensMoveState |
| 随机重启尝试次数 | 16n(n<<4) |
| 温度随时间的变化函数 | exp(start\_temp\_log - double(time) / (n << 5)) |
| 时间结束时的温度 | 1e-30 |
| 状态估值函数 | max\_conflicts - state.conflicts() |

1. 5秒内解出的最多皇后数n为：( 700 )
2. 简述除参数外对算法做出的修改（无修改则填无）：

start\_temp\_log设为 n/log\_n\*0.7，降低了初始温度

实验结果分析：



仍然观察到QueensMoveState在其它参数相同的情况下比QueensSwapState要快很多.随后调整初始温度和温度随时间的变化函数，实验证明在其他情况不变时，温度设为n/logn\*0.7的效果最好，0.8或0.6都会使算法 不断算出错误的解，使重启次数大大增加.随后又减慢了降温速率，以希望算法尝试更多的可能性，事实证明这一策略是有效的,将算法能够在五秒内解决的规模提升了20左右.最后降低结束时温度，因为n皇后的解在整个解空间中分布较均匀，每个初始点距离解并不远，因此并不需要大量的重启，真正重要的是在一个初始点的低温阶段微调，最终真正达到解的过程，低温阶段的"爬坡"行为能高效完成最后几步调整.若是最终温度设得比较高，很有可能在非常接近解得时候重启了，实际上造成了损失.从运行结果我们也可以看到，将终止温度设为1e-30只重启了一次就找到了解.5s对重启的敏感度是很高的，也因此，此参数尽量避免重启，就可以处理n接近700规模的问题.

**实验三：遗传算法求解n皇后问题**

实验目的：掌握遗传算法的思想与实现，探究影响遗传算法效果的因素

实验步骤：认真阅读作业要求说明文档中的说明，调整遗传算法的参数，尝试在5秒钟之内求解皇后数n尽可能大的n皇后问题（仅需输出1个解）

实验结果：

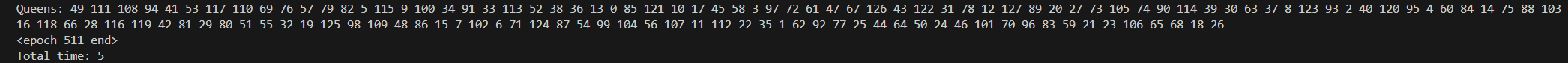
1. 参数设置情况

|  |  |
| --- | --- |
| **参数** | **参数取值** |
| 进化代数 | 8n |
| 种群大小 | 8n |
| 突变概率 | 0.9 |

1. 5秒内解出的最多皇后数n为：( 128 )
2. 简述除参数外对算法做出的修改（如基因编码、交叉、变异方式等，无修改则填无）

只输出Adaptability为0的解，不输出其他中间结果

实验结果分析：



经过实验，发现进化代数至少需要保证在8n或以上才能保证在大部分情况下找到解；种群大小在8n左右也比较适中，既不会使运算量特别大，也不会舍弃其他可能的方向；突变概率也设在0.9就比较合适，保证了种群基因的多样性，如此测试下来在n=120左右都能取得比较理想的效率.然而在n继续变大的时候，便需要更多的进化代数，于是时间开销会变大，很难在5s内完成求解.

**实验四：冲突最小化算法求解n皇后问题**

实验目的：掌握约束满足问题模型，掌握冲突最小化算法的思想与实现，探究影响冲突最小化算法效果的因素

实验步骤：认真阅读作业要求说明文档中的说明，调整冲突最小化算法的参数，尝试在5秒钟之内求解皇后数n尽可能大的n皇后问题（仅需输出1个解）

实验结果：

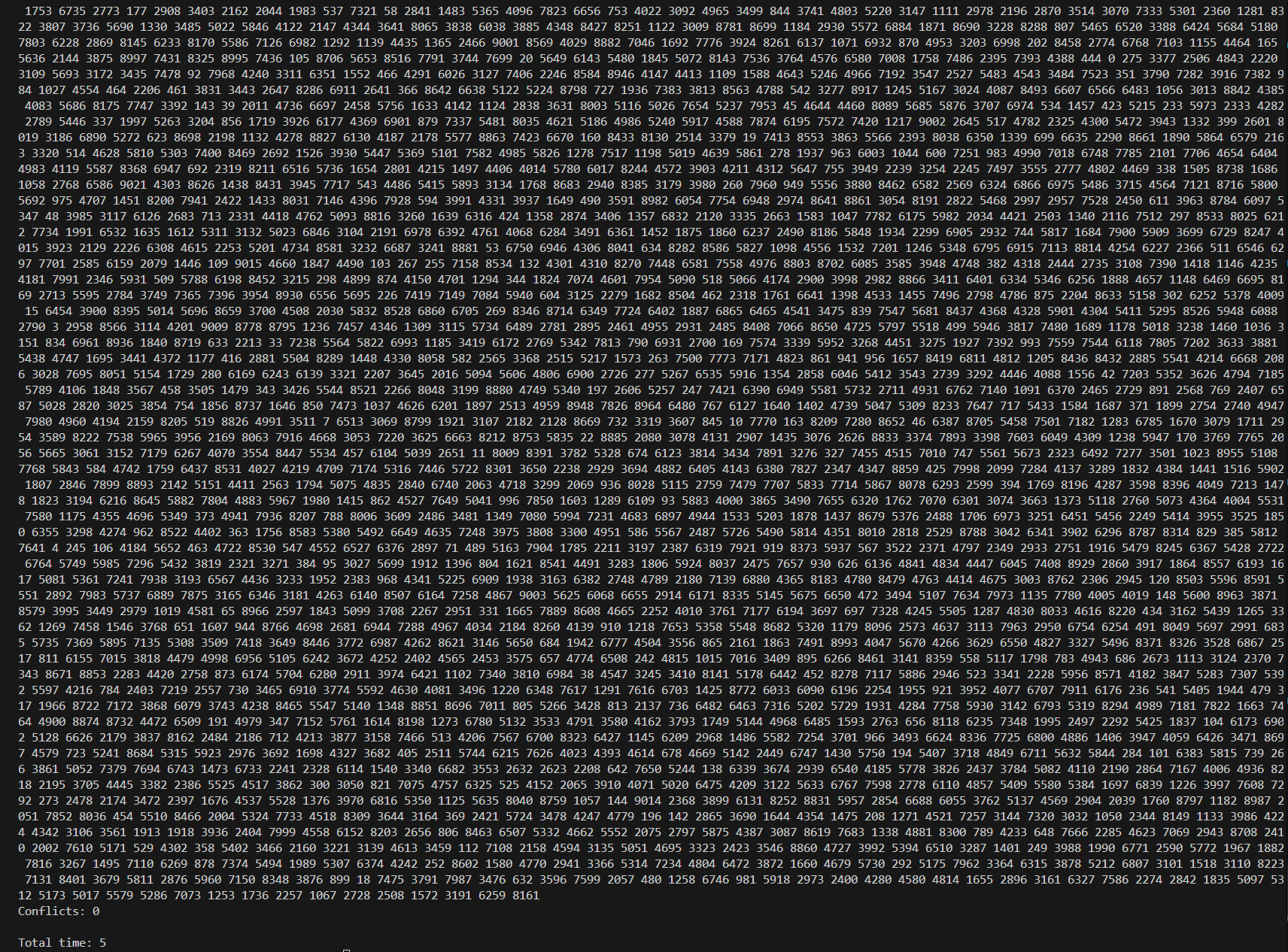
1. 参数设置情况

|  |  |
| --- | --- |
| **参数** | **参数取值** |
| 随机重启尝试轮数 | 10 |
| 单轮变元最大修改次数 | 4n |
| 选择算法 | Ms |
| 冲突数估值函数（若选择算法不是轮盘赌则不填此项） | / |

1. 5秒内解出的最多皇后数n为：( 9025 )
2. 简述除参数外对算法做出的修改（如选择待修改变元、待修改变元选择新值方式等，无修改则填无）

无

实验结果分析：



注意到此算法基本不需要重启，几乎每次都能一次搜索出结果，因此随机重启的次数并不重要；单轮变元最大修改次数设定在4n到16n之间的区别都不太大；选择算法上，由于n的规模很大，选择fbs或rs很可能导致收束不到解，经过实验，两者都无法实现和ms一样一次给出结果，因此选择了ms.结果显示，此算法可以完成远比前三种算法规模大得多的n皇后问题.

对于不限方法的求解，参见fast\_queens.cpp.