**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计**

**实验名称： 回溯法—地图填色问题**

**学院：计算机与软件学院 专业： 计算机科学与技术（创新班）**

**报告人： 何泽锋 学号： 2022150221 班级： 高性能特色班**

**指导教师： 杨烜**

**实验时间： 2024年4月30日**

**实验报告提交时间： 2024年5月1日**

**教务处制**

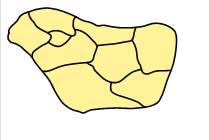
**一．实验目的**

1.掌握回溯法算法设计思想。

2.掌握地图填色问题的回溯法解法。

**二．问题描述**

1、对下面这个小规模数据，利用四色填色测试算法的正确性；



2、对附件中给定的地图数据填涂，请解释为什么填涂颜色超过了四色；

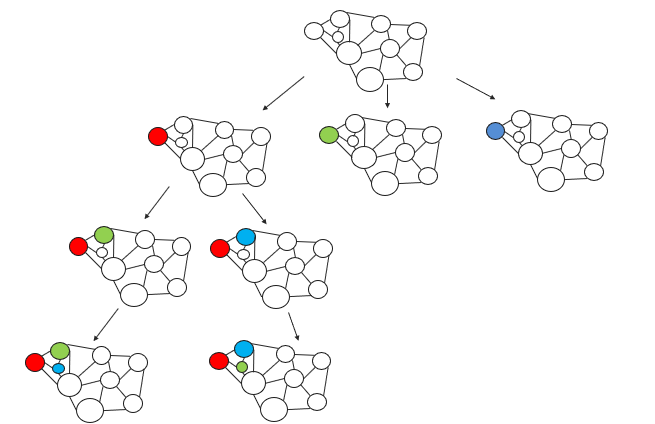
3、设计剪枝策略，分析不同的剪枝策略的效率。

4、随机产生不同规模的图，分析算法效率与图规模的关系

**三．算法原理**

**1.回溯法基本原理**

（1）基本原理：在包含问题的所有解的解空间树中，按照深度优先搜索的策略，从根结点出发深度探索解空间树。当探索到某一结点时，要先判断该结点是否包含问题的解，如果包含，就从该结点出发继续探索下去，如果该结点不包含问题的解，则逐层向其祖先结点回溯



（2）基本步骤

①明确问题的解空间树内容，此处为个点填涂的颜色

②确定结点的扩展搜索规则，在回溯法中，使用DFS。

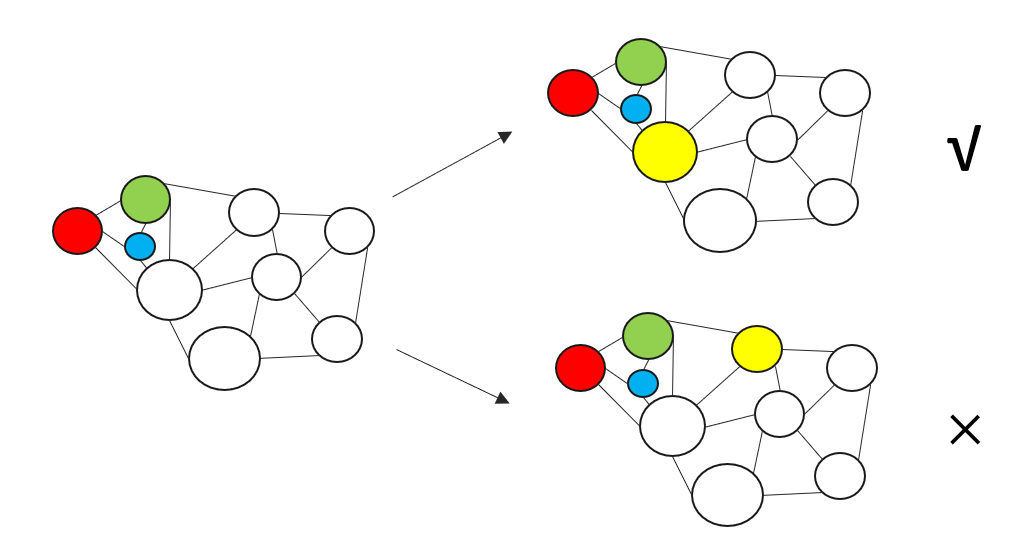
③构造约束条件。作为剪枝函数，避免无效搜索。

**2.优化剪枝**

（1）节点选择优化：

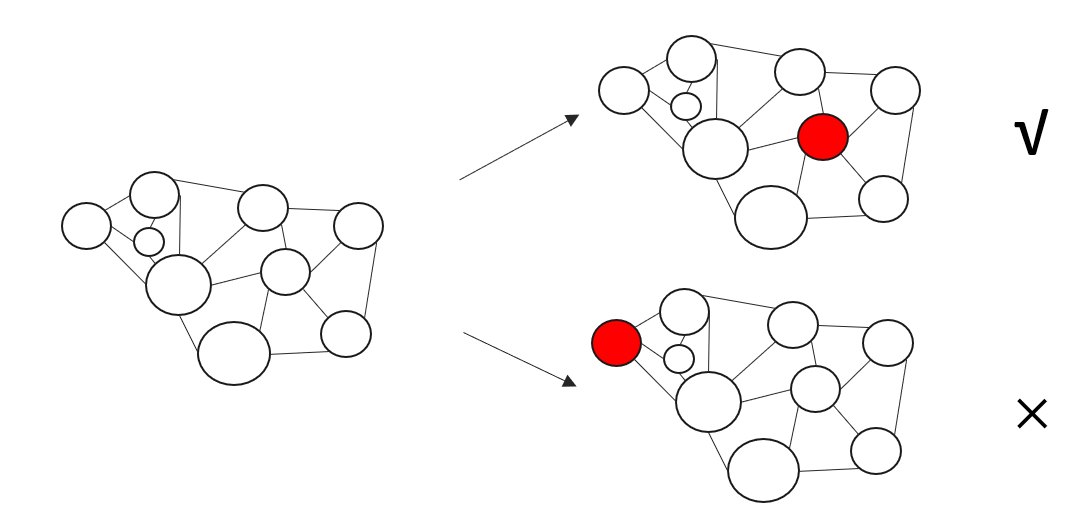
①最少剩余量选择原理（MRV）：最少剩余量选择是指优先选择剩余未填色节点中可填颜色最少的节点，通过此方法可以有效的减少搜索树中无法得到解的分支，从而增大路径可行解的几率

如图所示，下方的点比右侧的点可图颜色更少，若先涂右侧，会导致下方无色可填，但代码并不会立刻回溯，可能接着往右侧涂，但显然此时无法继续获得可行解，造成无效搜索



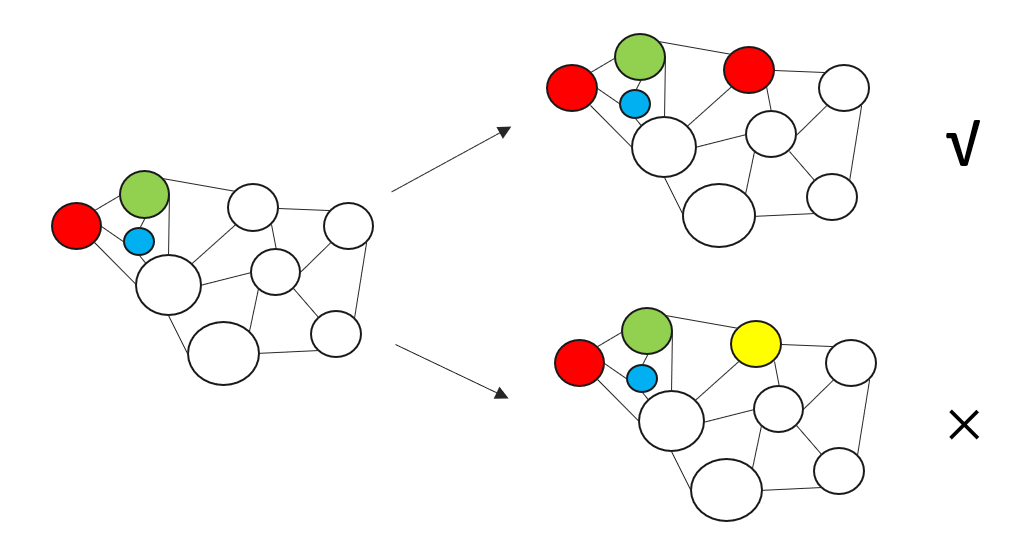
②最大度选择（DH）：优先选择度最大的节点进行填色，度数越大的节点受到其他节点的约束就越大，若先填涂其他节点容易导致度数大的节点无色可填，从而导致填涂无解情况，造成无效搜索。

如图所示，优先选择了周围节点最多的点涂色，即度最大的点。这是因为这个点受到的约束是最大的，若先填其周围，出现四种不同色时便无色可填，因此有必要优先填涂该点，确保有色可填



③最小约束值（LCV）：在可填颜色中，选择约束最小的颜色，即对相邻区域可选色影响最小的值，影响越小，周围点可涂方案会增加，越有可能获得可行解，此方法只在寻找单解的情况下有效，寻找全解或多解的情况下反而会影响程序运行时间

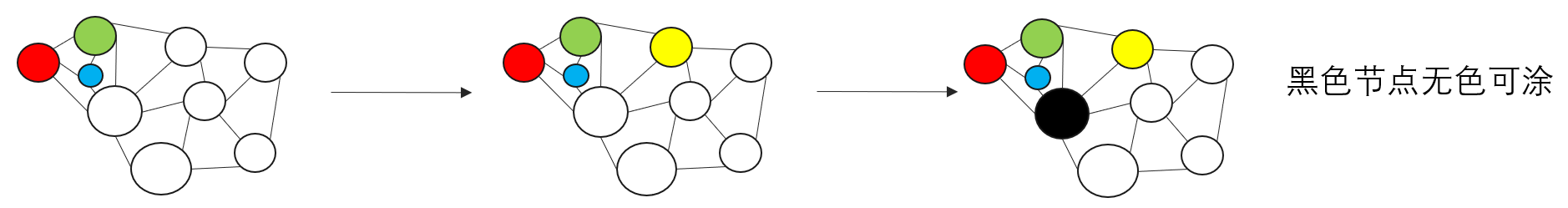
如图所示，右侧点有多种颜色可选，优秀选择对周围影响最小的可填色，这样相邻的节点有更多的选择。若选择其他颜色填涂，如下方填色方案可以看到，出现节点无色可涂，继续搜索造成时间浪费。



（2）涂色策略优化

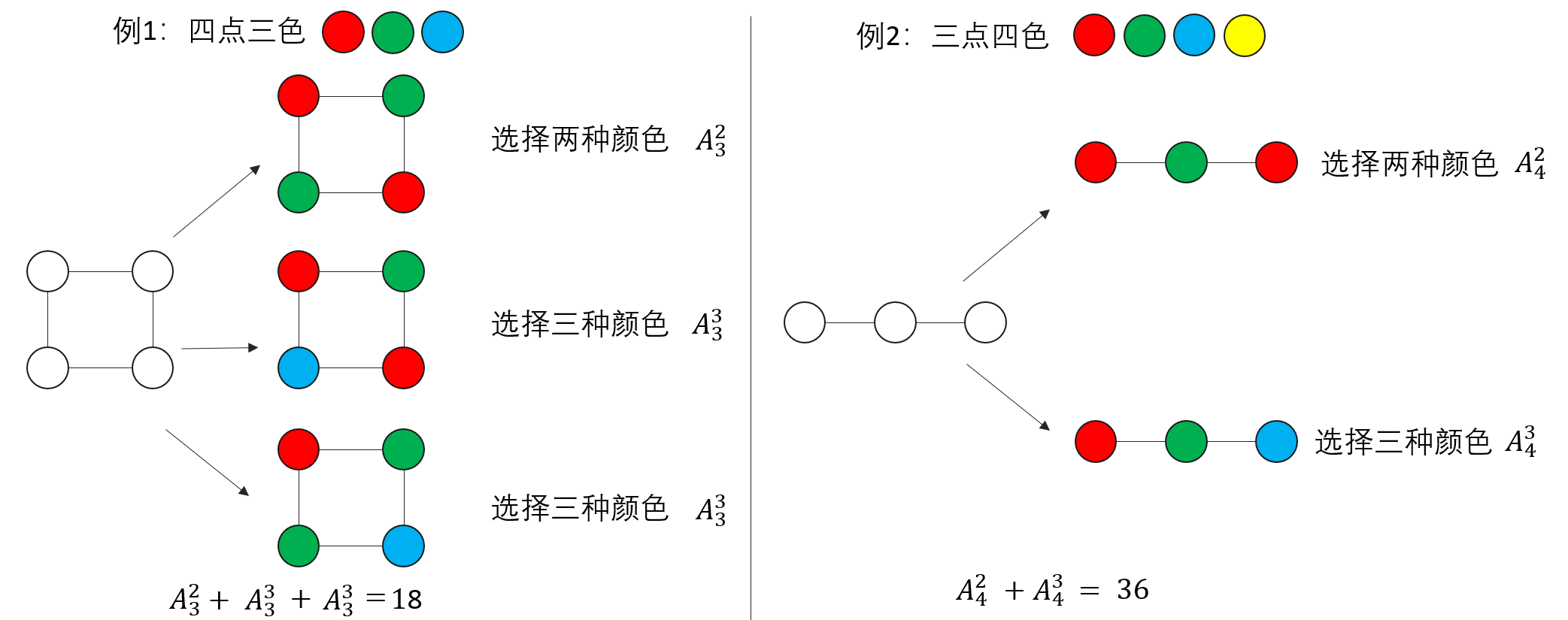
①向前检验：当前节点选择了一个颜色后，在相邻节点的可选颜色中将该颜色删去。如果在删除的过程中发现删除后没有颜色可以填涂，则说明当前节点不能涂此色，通过提前试错的方式，可以判断周围相邻点是否可填，从而提高了算法效率。

如图所示，当填完黄色节点后，其相邻黑色的节点出现无色可涂的情况，因此可知填黄色是不可能出现可行解的，可以提前回溯，避免无效的搜索。



②颜色轮寻：记录新颜色出现的首个节点，将同色的点视作一个分组，求解得到一组可行解后，按照选择的颜色数和总组数按照数学规律计算，得到其他的解。需要注意的是，使用颜色轮寻后，需要记录是否使用了新的颜色，出现新颜色需要记录此时使用的颜色总数。

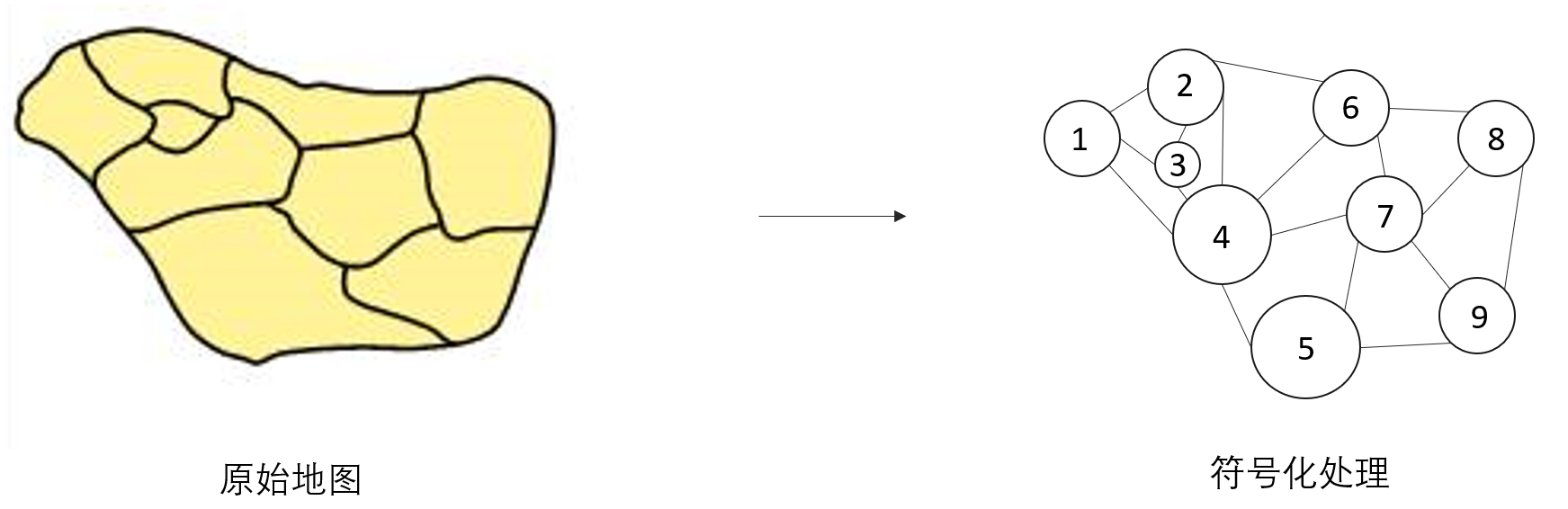
如图所示，填涂时可以不关心使用的具体是什么颜色，只需要知道到达该节点时是否有出现新的颜色，并且其与第一次出现该颜色的关系。具体来说，如例一所示，可以根据使用的颜色数，和第一次出现新颜色的位置来获得不同解，然后通过排列组合(使用的颜色数和总颜色数)计算得到同类型更改颜色位置的解。



**四. 数据结果**

**1.小规模数据涂色验证：**

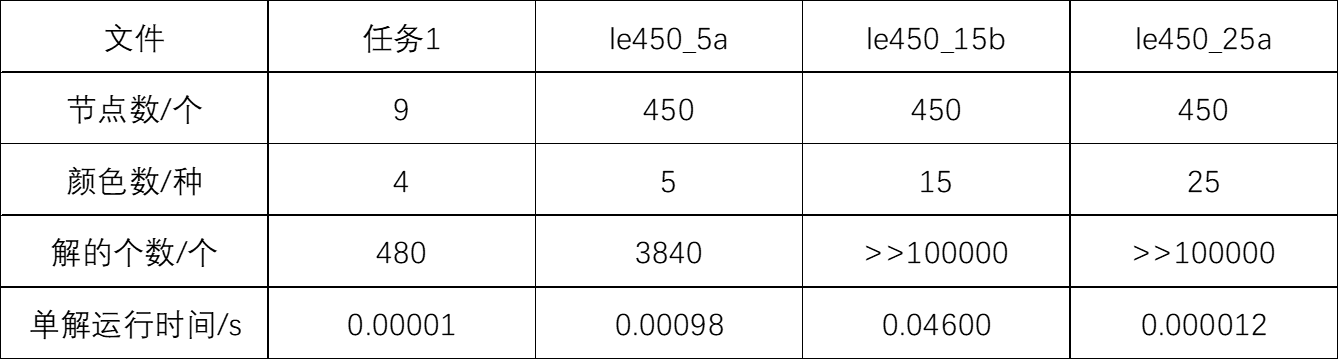
将实验给出的地图进行抽象化，将地图存入邻接矩阵中，运行剪枝后的程序最终得到480组解



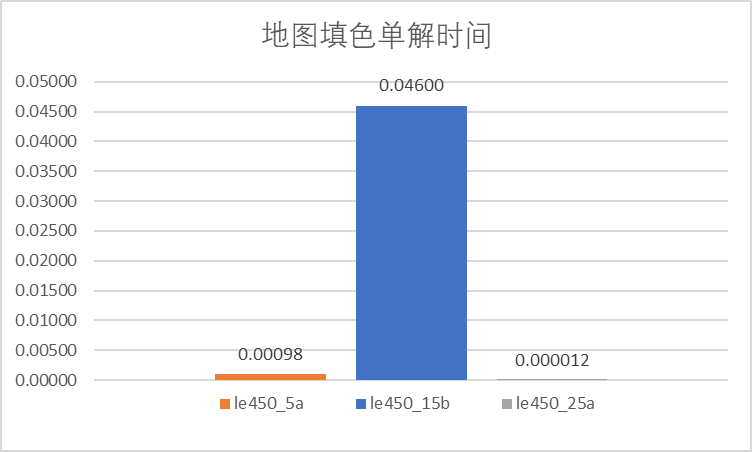


**2.使用剪枝后的程序对附件中的地图数据进行填写：**

首先需要读入文件，并处理无关信息，最终可得到三张地图的相关信息，需要注意的是附件中的地图最少填涂颜色分别为5、15、25，使用四色填涂无法获得可行解，这是因为这些地图都不是平面图，即抽象成节点后，相邻节点的连线存在交叉。将三组的运行结果和任务一的结果放在一起，可得到如下表格



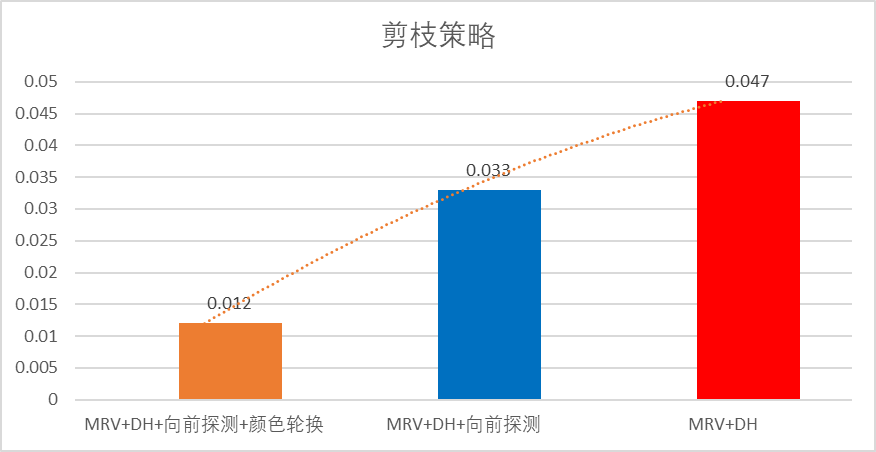
将附件中的三个地图单解运行时间做成图可以看到，15色的地图求单解所需时间最长，25色求解时间最短。需要注意的是三个图只有5色图可以求出有限个解（3840个），其余两图，解数过多，代码无法求解完全。



**3.分析不同剪枝策略的效率**

本次实验采用的剪枝策略包括：最少剩余量选择原理（MRV）、最大度选择（DH）、向前检验、颜色轮寻，验证采用了le450\_25a图。具体结果如下表所示：





分析：可以看到，单独采用MRV和DH无法得到可行解，结合两种方法后，求解10w解的时间为0.047s，添加向前探测后可以更快获得10w解。但使用颜色轮询后，此处时间为0.012，当求解数进一步加大时，使用颜色轮寻优化会更加明显，这是因为颜色轮寻是采用数学方法在获得单组解的情况下通过排列组合的知识得到其他的解，当可涂颜色数量较多的情况下，通过此方法可以获得阶乘数量的解，极大程度优化了时间。

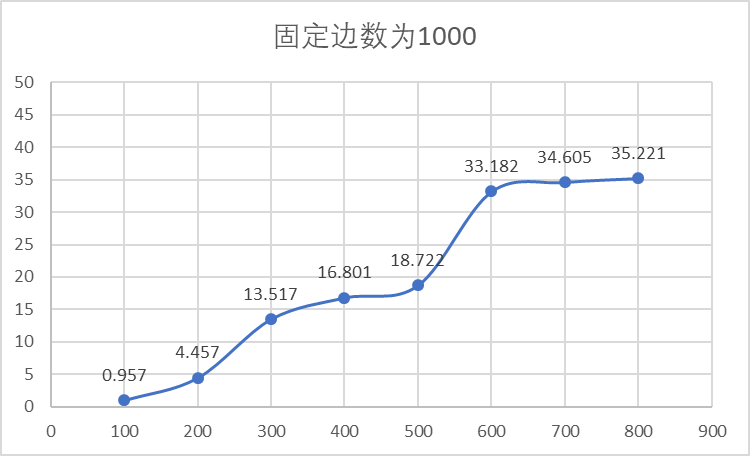
**4.随机生成不同规模的地图，分析算法效率与图规模的关系**

实验主要是验证了两种数据规模的变化，分别为固定边数增加点数和固定点数增加边数

①固定边的数量=1000 可填颜色数=10 解的数量=

实验得到数据如下表所示：

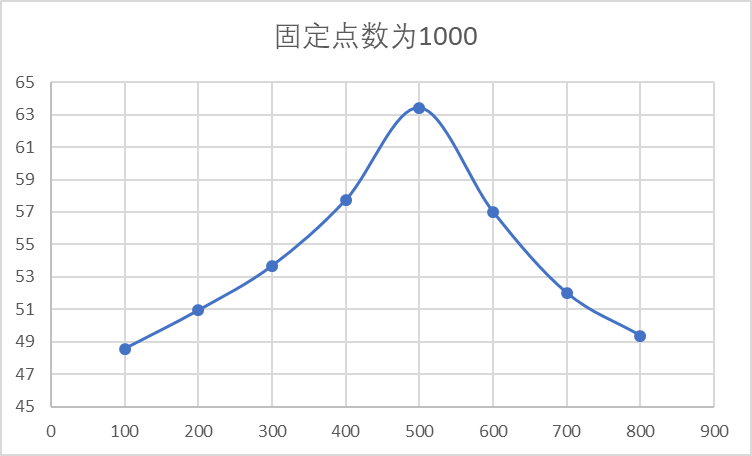




分析：随着点数的增加，求解的时间逐渐增加，这是因为边越多对邻点的限制越多，需要递归的深度加深，但不会呈阶乘的级别增加，原因是剪枝策略会提前终止不可能得到解的涂法，减少递归深度

②固定点的数量=1000 可填颜色数=10 解的数量=





分析：随着边数的增加，递归深度逐渐增加，而后当边数达到一定程度，边密度越大，图变得更加复杂，可选的颜色减少，算法剪枝的效率更高，所以搜索效率会更高。

**五．实验结论**

1.回溯法的复杂度为*O(n!)*，使用未剪枝的程序运行效率非常低，并且对于大部分数据无法在有限时间内求得可行解，需要使用剪枝优化。

2.本次实验使用的剪枝策略包括：最少剩余量选择原理（MRV）、最大度选择（DH）、向前检验、颜色轮寻。通过这几种剪枝策略可以极大的提高程序运行效率，尤其是使用了颜色轮寻可以在获得单组解的情况下扩展成颜色数量的阶乘级别数量的解。

3.对附件中的地图填色使用的颜色超过四色，这是因为附件中的地图不是平面图，节点之间会出现交叉的情况，因此其维度产生了变化，对周围色的影响更多，需要使用更多的颜色才能找到可行解，这也与图中的完全子图有关，完全子图的阶数越高需要的颜色也就越多。同理对于随机生成的地图无法控制其为平面图，因此可涂颜色设置为10色，一定程度上提高了获得可行解的可能。此处统一获取10亿组解是因为使用颜色轮寻后可以很快的扩展组的数量，因此需要较大规模数据才能获得较稳定的运行时间。

|  |
| --- |
| 指导教师批阅意见：    成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。