# 最大团问题的分支限界算法

## 最大团问题的定义

最大团问题是指在一个简单无向图G = (V, E)中寻找顶点最多的一个子图C，使得。在本文中，我们使用下列定义：

- 补图——给定一个图G(V, E)，它的补图定义为 (V, )，其中。

- 极大团——图的一个团，该团不能通过添加图的顶点而扩大。

- 独立集——图G = (V,E)的子图S，其中。显然，图的最大团是其补图的最大独立集。

- 候选顶点——可以用来扩充当前团的顶点，即与当前团所有顶点都邻接的顶点。

- 颜色数——在着色过程中赋予顶点的自然数。

- 染色数——对图进行着色所需的最小颜色数。

分支限界算法是求解最大团问题最常用的算法。算法步骤一般分为分支部分与限界部分，其中分支部分选择候选顶点集中的一个顶点进行分支，将该顶点添加到当前团中，在新的节点（新的递归过程）中，候选集被更新为原候选集中与该顶点邻接的所有顶点，因此候选集中所有的顶点都邻接于当前团的所有顶点。而限界部分计算当前候选集中，最大团的大小的上界，如果该上界加上当前团的顶点数不大于目前找到的最大的团，则该分支不可能产生更好的团，因此需要裁减该分支。

本文主要工作在于研究目前最先进的几个分支限界算法，并在这个基础上，构造出效率更好的求解算法。

## 下界与代价函数

分支限界法算法中，分支部分保证了算法的完整性与精确性，而限界部分则决定了算法的效率。为了提高算法效率，关键在于估算出尽可能紧的代价函数，即团大小的上界，以及尽快找到一个大的下界，即一个尽可能大的团。

估计代价函数时，一个平凡的做法是以候选集的顶点个数作为代价函数，如果当前团大小加上候选集顶点个数不超过下界，则可以当前分支无需进入下一轮迭代。然而，这样的代价函数过于松散，未能精确估计团的上界，导致算法效率不高。

### 2.1 染色数

图的染色数可以作为最大团问题的上界（命题1）。

**命题1** (Balas 和Yu 1986)[] 任意图的最大团大小不大于该图的染色数。

这个命题可以从以下事实直接得出：一个具有k 个顶点的团至少需要用k 种颜色进行着色，因为这些顶点都是两两邻接的，因而团的任意两个顶点不能具有同一种颜色。

### 2.2 贪心着色

染色数问题所对应的判定问题，本身就属于NP难问题，因此使用染色数作为最大团问题的代价函数是不实际的。另一种方案则是使用成为贪心着色[]的方法，求出染色数的近似解，作为最大团问题的代价函数。贪心着色过程遵循如下原则：

- 如果(v1,，v2)∈E，则No[v1] ≠ No[v2]

- No[v] = 1，或者No[v] = k > 1，则存在顶点v1∈NR[v]，v2∈NR[v]，...，vk-1∈NR[v]，使得No[v1] = 1，No[v2] = 2，....，No[vk-1] = k-1。

其中，No[v]表示顶点v的颜色编号，NR[v]表示顶点v在子图R中的邻居。

显然，贪心着色过程所用到的颜色数可作为最大团问题的代价函数。对于顶点个数为n的图，贪心着色过程的时间复杂度为O(n2)。

### 2.3下界与分支顺序

好的分支顺序能够使算法快速更新下界，从而在后续迭代中得以裁剪更多的分支。贪心着色过程为分支顺序做了很好的准备——优先选择颜色编码大的顶点。因为团最多只能在每个颜色编码中选取其中一个顶点，而颜色编码最大的顶点，在所有其他颜色编码中都存在与之相邻的顶点。另外，这种分支顺序使得在迭代树的同一层中，只需要进行一次贪心着色过程，因为排除了所选择的顶点后，新一轮贪心着色过程不会产生新的着色方案。

### 2.4 顶点顺序对着色的影响

贪心着色过程根据既定的顺序，依次对顶点进行着色。不同的顺序对着色效果（即所拥得颜色数）有很大的影响。直觉上，限制越多的顶点应该优先进行着色，因此，如果只考虑上界问题的话，对顶点按照度数大小降序排列，可以得到较紧的上界。这便是MCQ[]算法所采用策略。

然而，正如前面所述，贪心着色过程在分支限界算法中，不仅仅是获取上界的手段，同时也对分支策略有重大的影响。顶点被分支的次序与其顶点的颜色值大小顺序一致，且在同一种颜色值的顶点中，越晚被着色的顶点越早被选择，因此，这里需要考虑新的排序准则[]。

将顶点排列成如下顺序V={V[1],V[2],...,V[n]}，使得在图G=(V, E)由顶点V’ = {V[1], V[2], ..., V[i]} (i ≤ n) 导出的子图中，使V[i]始终都是V’中度数最小的顶点。如果{V[1], V[2], ..., V[i]}中的所有顶点都有着同样的（最小的）度数，也就是说由{V[1],V[2],...,V[i]}导出的子图是正则的，则对这些顶点进行排序时没有意义的。在这种情况下，我们直接将这些顶点放置在顶点集中。虽然这个排序过程是比较耗时的操作，但是该操作只在算法初始化的时候进行，所以从全局来看，负担是相当小的。

由此便得到了MCR算法[]，该算法是MCQ算法的改进版，效率有了明显的提升。

值得注意的是，虽然在分支限界算法框架下，迭代树的同一层只需进行一次着色，然而，在不同层时，需要对新产生的候选顶点集进行贪心着色过程。贪心着色过程对顶点顺序具有部分的继承性。当着色完成后，颜色编号相同的顶点之间，原来的顺序并没有因着色而被破坏，先着色的顶点仍然在后着色的顶点之前。这样就使得下层迭代时，顶点是相对有序的。但是，这种顺序会在逐次迭代中失去，所以仅仅在初始阶段对顶点进行排序是不够的。考虑到排序操作的时间复杂度较高，必须权衡排序本身的效率负担与因排序所得的效率提升。下面分别介绍两种处理策略。

#### 2.4.1 静态顺序顶点辅助集

为了延续初始阶段顶点顺序的效果，在初始化之后，使用一个全局的顶点辅助集保存顶点的顺序。当候选顶点集需要重新着色时，根据顶点辅助集中的顺序，对候选集中的顶点进行着色。由于顶点辅助集保存了初始顶点顺序之后不再更改，因此称之为静态顺序顶点辅助集。虽然静态顺序与预期的候选集顺序不完全一致，但是作为一种近似估计，顶点辅助集较好地平衡了排序的耗费与收益。这便是MCS所采取的策略。MCS相比MCR增加了静态顺序顶点辅助集，以及后面即将提及的重着色技术。

#### 2.4.2 动态排序策略

相比静态顺序一经确定便不再更改的策略，动态排序策略在特定的时刻对顶点进行重新排序，使其更加适应当时的候选集。由于没有辅助顶点集，所以需要使用一些技巧来增强贪心着色过程对顶点顺序的继承性。

### 重着色

## 三、动态排序

## 四、Maxsat推理