一、基本描述

类似于回溯法，也是一种在问题的解空间树T上搜索问题解的算法。但在一般情况下，分支限界法与回溯法的求解目标不同。**回溯法**的求解目标是找出T中满足约束条件的**所有解**，而**分支限界法**的求解目标则是找出**满足约束条件的一个解**，或是在满足约束条件的解中找出使某一目标函数值达到**极大或极小的解**，即在某种意义下的**最优解**。

（1）分支搜索算法

所谓“分支”就是采用广度优先的策略，依次搜索E-结点的所有分支，也就是所有相邻结点，抛弃不满足约束条件的结点，其余结点加入活结点表。然后从表中选择一个结点作为下一个E-结点，继续搜索。

选择下一个E-结点的方式不同，则会有几种不同的分支搜索方式。

1）FIFO搜索

2）LIFO搜索

3）优先队列式搜索

（2）分支限界搜索算法

二、分支限界法的一般过程

由于求解目标不同，导致分支限界法与回溯法在解空间树T上的搜索方式也不相同。**回溯法以深度优先的方式搜索解空间树T**，而**分支限界法则以广度优先或以最小耗费优先的方式搜索解空间树T。**

分支限界法的**搜索策略是**：在扩展结点处，先生成其所有的儿子结点（分支），然后再从当前的活结点表中选择下一个扩展对点。为了有效地选择下一扩展结点，以加速搜索的进程，在每一活结点处，计算一个函数值（限界），并根据这些已计算出的函数值，从当前活结点表中选择一个最有利的结点作为扩展结点，使搜索朝着解空间树上有最优解的分支推进，以便尽快地找出一个最优解。

分支限界法常以广度优先或以最小耗费（最大效益）优先的方式搜索问题的解空间树。问题的**解空间树是表示问题解空间的一棵有序树，常见的有子集树和排列树。**在搜索问题的解空间树时，分支限界法与回溯法对当前扩展结点所使用的扩展方式不同。在分支限界法中，每一个活结点只有一次机会成为扩展结点。活结点一旦成为扩展结点，就一次性产生其所有儿子结点。在这些儿子结点中，那些导致不可行解或导致非最优解的儿子结点被舍弃，其余儿子结点被子加入活结点表中。此后，从活结点表中取下一结点成为当前扩展结点，并重复上述结点扩展过程。这个过程一直持续到找到所求的解或活结点表为空时为止。

三、回溯法和分支限界法的一些区别

有一些问题其实无论用回溯法还是分支限界法都可以得到很好的解决，但是另外一些则不然。也许我们需要具体一些的分析——到底何时使用分支限界而何时使用回溯呢?

回溯法和分支限界法的一些区别：

方法对解空间树的搜索方式　　存储结点的常用数据结构　　结点存储特性常用应用

回溯法深度优先搜索堆栈活结点的所有可行子结点被遍历后才被从栈中弹出找出满足约束条件的所有解

分支限界法广度优先或最小消耗优先搜索队列、优先队列每个结点只有一次成为活结点的机会找出满足约束条件的一个解或特定意义下的最优解