# 分支限界法和回溯法的异同

## 同

分支限界法类似回溯法：都是在问题的解空间（人工智能中把它叫做状态空间）上搜索问题的解

## 异

### 求解目标不同：

分支限界法的求解目标是找到满足约束条件的一个解，有时需要找到最优解。

回溯法的求解目标是找到满足约束条件的所有解，因此回溯法在搜索时采用深度优先搜索，因为从根节点开始，如果能够搜索到叶结点，那么就得到了问题的一个解。

### 搜索方式不同

（显然搜索方式的不同是由两者求解目标的差异决定的）

分支限界法以广度优先或最小耗费优先的方式搜索问题的解空间树

显然回溯法的搜索方向是纵向(深度）的，而分支限界法的搜索方向是横向（广度）的

### 最优解的构造方法不同

在回溯法中，我们也需要处理这个问题：对于子集树问题，我们通常采用的方法是维护一个数组x[n]，数组的元素x[i]取值为0或1，表示问题中的第i个对象是否选择到最终解。

在队列式分支限界法中，由于搜索的方向不是纵深的，而是横向的，因此最优解的构造过程不像前者那样是连续的，逐个累积的；相反，这种搜索方式的最优解构造过程是离散的，非连续的，并且最优解的出现是难以预测的，每个当前扩展结点都有可能成为最优解的一部分。因此在程序中需要时时刻刻记住当前这个结点是从哪个结点过来的，即每个扩展结点需要知道它的父母结点。

因此造成回溯法和分支限界法最优解的构造的方法不同的根本原因在于：两种方法搜索的方向不同。

因此在分支限界法中普遍采用的方法是：定义结点。

# 广度优先与最小耗费优先对比

## 1.广度优先

活结点表采用的数据结构是队列，显然这种方法无需计算优先级，或者说活结点表中结点的优先级是由结点到达队列的先后次序决定的。然后根据优先级来选择下一个作为扩展结点的结点。活结点一旦成为扩展结点就一次性产生其所有儿子结点，这种方法也舍弃导致不可行解或导致非最优解的儿子结点

## 最小耗费优先

活结点表采用的数据结构是优先队列，显然这种方法是需要计算优先级的。每个结点的优先级由优先级函数来计算得出。然后根据优先级来选择下一个作为扩展结点的结点。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，会舍弃导致不可行解或导致非最优解的儿子结点

考虑如下0-1背包问题的实例：

n=3, c=30, w=[16,15,15], p=[45,25,25]

1

0

队列式分支限界法：[]表示队列

[A]B,C>BC

[BC] DE >E

[CE]FG>FG

[EFG]JK>K(45)[1,0,0]

[FG]LM>L(50)[0,1,1]M(25)

[G]NO>N(25)O(0)

* 优先队列式分支限界法：

优先队列的优先级定义为活结点所获得的价值。

[A]BC>B(45)C(0)

[BC]DE>E(45)

[EC]JK>K(45)[100]

[C]FG>F(25)G(0)

[FG]LM>L(50)[0,1,1]M(25)[0,1,0]

[G]NO>N(25)[0,0,1]O(0)[0,0,0]

.