5-2最小长度电路板排列问题

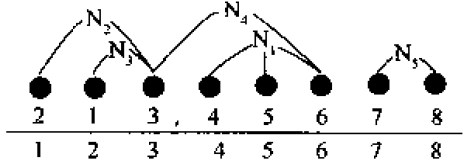
问题描述：最小长度电路板排列问题是大规模电子系统设计中提出的实际问题。该问题的提法是，将n 块电路板以最佳排列方案插入带有n 个插槽的机箱中。n 块电路板的不同排列方式对应与不同的电路板插入方式。

设B={1,2,···，n}是n 块电路板的集合。集合L={N1，N2，···，Nm}是n 块电路板的m 个连接块。其中，每个连接块Ni 是B 的一个子集，且Ni 中的电路板用同一根导线连接在一起。

在最小长度电路板排列问题中，连接块的长度是指该连接块中第一块电路板到最后一块电路板之间的距离；

试设计一个回溯法找出所给的n个电路板的最佳排列，使得m个连接块中最大长度达到最小。

**问题框架：**



在最小长度电路板排列问题中，连接块的长度是指该连接块中第1块电路板到最后1块电路板之间的距离。例如在图示的电路板排列中，连接块N4的第1块电路板在插槽3 中，它的最后1块电路板在插槽6中，因此N4的长度为3。同理N2的长度为2。图中连接块最大长度为3。试找出所给n个电路板的最佳排列，使得m个连接块中，最大长度达到最小。

**问题理解：**

该问题是一颗排列树，将电路板按不同的顺序排列，会导致连接块之间的最小长度发生变化。

解向量：x=()，的取值为1,2,…n,表示第i个位置放置的电路板的编号。

**目标函数**：

其中，表示第k个连接块内部的最大长度，是获取不同连接块的最大长度的最大值， m表示连接块的数量。

**剪枝函数**：

这是个最小化问题，因此需要估计目标函数的下界，假设当前搜索到了，则对的下界估计为

即考虑{}的连线长度，未搜索到的情况都认为是0

上面对于未搜索到的情况都按照0来处理，**即对于不确定的情况取最小值，**这样才不会把潜在的可行解排除掉。还有一种更紧的下界估计，主要是针对未搜索到的情况进行分析，得到比0大的电路板长度的估计。

上式中第一种情况是全部在遍历过的插槽里，这时的结果上之前一样，第二种情况是全部在未遍历的插槽里，这时最少的长度和本身包含的电路板数量一样的，第三种情况是包含的电路板跨越了当前插槽，这时的最理想的安排方式是把剩余的节点紧密地排列在后面。

如果当前的最大长度小于小于最优最大长度，则继续搜索，否则回溯.

**5-6无和集问题**

问题描述：

设 S 是正整数集合。 S 是一个无和集，当且仅x ,y 属于 S, 且x +y 不属于 S。 对于任意正整数k，如果可将 { 1,2,...,k}划分为 n 个无和子集 S1、S2、.........、Sn，称正整 k 是 n可分的。记 F(n)=max{ k | k 是 n可分的}。试设计一个算法，对任意给定的 n，计算 F (n )的 值。

**问题理解**：

对于任意给定的n,F(n)表示最大的n可分的正整数，也就是说，(1,2,….,F(n))这n个正整数可以划分成n个无和子集。因此，解向量x=()，其中的取值范围是{1,2,…n}表示整数i被分配到哪个集合。

对从1到k的正整数，需要将其中的每一个元素分散到固定的n个子集中，因此，解空间是一颗n叉子集树，第m层的n个分支表示将第m个元素（即m本身）分配到n个子集中的一个。

N叉子集树的搜索复杂度是O()

**1 2 3 4 5 6 7 8 9……k**

**n1 n2 n3**

**固定n的个数，求最大的k**

**剪枝函数**：

1. 可行性约束：当前节点一共有n个子节点，如果选择第i个节点，表示树的深度deep对应的值放入到集合i中。允许放入的条件是：集合i中已有的任意两个元素之和不等于deep,否则进行剪枝.
2. 边界约束：对于当前的深度deep，如果遍历了所有的子集分配方案，都不合适，即整数deep被所有n个集合都拒绝时，F(n)就找到了，为deep-1,程序结束。这一约束条件的合理性在于{1,2,…k+1}所构建的子集树是在{1,2,…k}子集树的基础上增加了一层，如果后者所有的叶子节点都是不能拓展的，则层数更大的树一定也不是可行解。
3. 搜索深度约束：当n很大时，搜索很耗时，因此需要给定一个运行时间约束。

**算法描述：**

1. 初始化搜索时间t=0,构建n个集合，树的深度deep=1，deep=1放入第一个集合
2. Deep = deep +1
3. 为deep值分配集合，从集合1开始，若满足约束则放入，否则回溯，若deep无法放入任何集合，则此时F(n)获得，是deep-1
4. 判断是否超时，未超时回到2）,否则结束程序

**5-13 工作分配问题**

**问题描述：**

设有n件工作分配给n个工人。将工作i分配给第j个工人所需的费用是,试设计一个算法，为每一个人都分配1件不同的工作，并使总费用达到最小。

**问题理解：**

该问题是一个排列树问题。

解向量定义：x=()，其中的取值为{1,2,….n}表示第i项工作，被分配给的工人编号。

解空间树为排列树，遍历复杂度为O(n!)

**剪枝函数**：

下界约束：当前的最小费用为C,如果当前节点加入到集合之后的费用，则进行剪枝。

一个更紧的下界：当前节点的费用之和+未分配任务的最小费用之和

**算法流程**：

当 t = n时，n件工作全部完成，且当前的费用一定优于上一个最优费用，更新best price.

当 t < n时，工作未分配完，根据剪枝函数确定是否将任务分配给当前工人。

复杂度分析：

对于每一个节点都需要计算此时的总费用，如果从根节点开始计算，时间复杂度为O(n)，如果对历史结果进行保存，则每个节点的计算复杂度为O(1),排列树总共有n!个节点，因此，总的时间复杂度为O(n\*n!)或O(n!)