分支与定界算法的实现研究

李 胜 华*

(湖北大学数学计算机科学学院，湖北 武汉 430062)

摘 要：本文介绍分支与定界算法的基本原理，着重讨论了此算法在具体实现过程中遇到的两个难点：分支结点的寻找和当前结点对应数据的生成。

关键词：分支；定界；叶结点；当前结点

中图分类号：TP301.6，O221.4　文献标识码：A　文章编号：1671－1785（2003）02－0021－03

1 引言

分支与定界算法已成功地应用于求解整数规划问题、生产进度表问题、货郎担问题、选址问题、背包问题以及可行解的数目为有限的许多其它问题。对于不同的问题，分支与定界的步骤和内容可能不同，但基本原理是一样的。在具体实现过程中，用搜索树来描述其求解过程，对搜索树上的每个结点对应的可行解集计算一个下界或上界，利用界限选择分支结点，分支产生的结点再定界，直至产生最佳解。所以处理分支结点是主要工作。那么选择什么样的数据结构来表示结点及如何表示结点对应的数据，以利于分支结点的寻找及其子结点的定界就显得非常有意义。本文将重点讨论在分支与定界算法实现过程中，分支结点的寻找以及当前结点对应的数据如何生成两个问题。

2 分支与定界算法

分支与定界算法的基本思想是对有约束条件的最优化问题的所有可行解(其数目为有限)空间适当地搜索。我们约定本文涉及的问题为最小值问题。具体执行时，把全部可行解空间不断分割为越来越小的子集（即分支），并且为每个子集内的解的值计算一个下界（即定界）。每次定界后，把搜索树上当前所有叶子结点的下界比较，找出下界最小的结点，此结点即为下次分支的结点。逐渐分支必定会找到可行解，将目前已知最好的可行解及其值存放起来，如果待分支的结点的下界小于存放的值，则继续分支，否则算法终止，存放的解即为最佳解。

3 分支结点的寻找

分支与定界算法实施过程中，每次总是挑选下界最小的叶子结点作为下一次分支的结点。鉴于叶子结点处理的频繁性，我们应该建立搜索树的叶子链。该分支的结点就不是叶子结点了，而它的子结点为叶子结点，所以对叶子链的操作为删除一个元素和增加若干个元素。为了便于处理，这个叶子链应为有序的。相应结点的数据类型及算法如下：

```
typedef struct TNode {
        BoundType bound; //BoundType 为限界的类型
        Struct Tnode*Childs [n]; //n 为子结点的个数，即分支数
        Struct Tnode*nextl; //指向下一个叶子
}*SearchTree;
```

收稿日期：2001—12—27

作者简介：李胜华（1972—），女，理学硕士，湖北大学数学与计算机科学学院讲师。

```
Status Search _Branch (SearchTree&L) {
        //带头结点的叶子链L 中删除第1个元素，并且将这个元素的子结点有序地插入叶
        子链中
        p=l->nextl;
        l->nextl=p->nextl; //删除第1个元素
        for (i=0; i++) {
                        q=p->childs[i];
                        r=l;
                        s=l->nextl;
                        while (s && q->bound>s->bound) {
                              //用s 指向插入的位置，r 指向其前趋
                              r=s;
                              s=s->nextl;
                              }
                        r->nextl=q;
                        q->nextl=s; //插入第i 个子结点
                        }
} //Search _Brach
```

4 当前结点对应的数据生成

搜索树中每个结点对应可行解的一个子集，我们可以用一些数据来描述这些解的特征，用来计算结点对应的下界。子结点对应的数据可由父结点对应的数据及分支规则来生成。由于每个叶子结点都有可能成为待分支的结点（即当前结点），故叶子结点对应的数据都应保留。但当数据量很大时，每个结点保留这些数据空间过于浪费。此时，我们可以采用另外一种办法，临时生成当前结点对应的数据。下面以货郎担问题为例介绍这种方法。

货郎担问题求解过程中，每个结点对应一个归约矩阵，用来计算下界，把搜索树中每个结点的归约矩阵存放起来，需不少的空间。由于此问题是按照是否包含某条边来二分解的集合，故每个结点对应的解集特征可用两个集合来描述：一定包含的边集E1和一定不包含的边集E2。这两个集合应该存储起来，如果一个结点对应的解为最佳解，则其E1为最佳路线，并且E1和E2可以用来产生当前结点对应的费用矩阵。我们现在只需利用E1和E2的信息来修改原费用矩阵。对于E1中的每条边 (i, j)，以后不能挑以i 为始点或以j 为终点的边，且有可能和E1中的路径连成回路的边也不能选，对于E2中的边，以后都不能选，故将原费用矩阵这些边的费用改为∞，然后归约即可，为此使用下面的数据类型：

```
typedef struct Tedges {
                        int  i;
                        int  j;
                        struct *nexte//指向下一条
}*Edges;
typedef struct TNode {
                        BoundType bound;
                        Edges E1, E2; //E1和E2分别为包含的边链及不包含的边链
                        Struct Tnode *childs[2];
                        Struct Tnode *nextl;
```

}*SearchTree；

5 结束语

虽然对于用分支与定界算法解决整数规则等问题在运筹学里有一套完整的理论，但将它用计算机来实现还是有一定的难度。本文正是在这个方面作了一些探讨，提出了适合分支结点寻找的存储结构及相关算法，并且以货郎担问题为例给了当前结点对应数据生成的一种方法。当然在这里面还有些算法有待细化，需进一步研究探讨。

【参 考 文 献】

[1] 王燕来，陈宝林，马仲蕃等.运筹学与最优化理论卷[M].北京：清华大学出版社，1998. 201－208.
[2] 卢开澄.组合数学算法分析[M].北京：清华大学出版社，1988. 58－69.
[3] 周培德.算法设计与分析[M].北京：机械工业出版社，1996. 46－51.
[4] 严蔚敏，吴伟民.数据结构[M].北京：清华大学出版社，1997.
[5] 李智渊.语言[M].成都：电子科技大学出版社，1988.

Research on the Implementation of Branching-Bounding Algorithm

LI Sheng-hua

(College of Mathematics and Computer, Hubei University, Wuhan Hubei, 430062, China)

Abstract: This paper introduces the basic principles of the branching-bounding algorithem. It mainly discusses two difficult points during the implementation of theis algorithm. They are the searching of the branching node and the replace of the data of the active node.

Key words: branching; bounding; leafnode; active node