列生成column generation

目录

1.1 修正单纯形法 Revised Simplex method

参考 Introdution to Column generation 例子 [1]

$$
\begin{aligned}
min \quad & z = 24x_1 + 29x_2 + 10x_3 + 38x_4 \\
s.t. \quad & x_1 + 4x_2 + 5x_3 = 60 \\
& 2x_2 + 3x_3 \leq 12 \\
& 2x_1 + x_2 - x_3 + 4x_4 \geq 10 \\
& x_i \geq 0
\end{aligned}
$$

添加松弛变量，使不等式约束变成等式约束

$$
\begin{aligned}
min \quad & z = 24x_1 + 29x_2 + 10x_3 + 38x_4 \\
s.t. \quad & x_1 + 4x_2 + 5x_3 = 60 \\
& 2x_2 + 3x_3 + x_5 = 12 \\
& 2x_1 + x_2 - x_3 + 4x_4 - x_6 = 10 \\
& x_i \geq 0
\end{aligned}
$$

转化为矩阵形式

$$
\begin{aligned}
min \quad & z = c^t x = [c_B \quad c_N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} \\
s.t. \quad & Ax = [B \quad N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b \\
& x \geq 0
\end{aligned}
$$

最终结果: 表1

变量	值	检验数
x_B	≥ 0	0
x_N	$= 0$	≥ 0

找不到进基变量，单纯形法（找到最优解）停止。
另外一种是无界停止，下面的步骤会说明。

step 1： 选择基矩阵 B 和基变量 x_B
不妨选择(x_1, x_3, x_5)，保证基矩阵可逆即可。

$$
Bx_B = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 1 & 1 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} 60 \\ 12 \\ 10 \end{bmatrix}
$$

$$\bar{x}_B = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix}$$

step 2： 计算对偶解

$$\pi B = [\pi_1 \quad \pi_2 \quad \pi_3] \begin{bmatrix} 1 & 5 & 0 \\ 0 & 1 & 1 \\ 2 & -1 & 0 \end{bmatrix} = c_B = [24 \quad 10 \quad 0]$$

求解结果：

$$\pi = [4 \quad 0 \quad 10]$$

step 3： 定价（找进基变量）

reduced cost=$c_j - \pi A_j$

对于基变量的检验数为0，(min) 找非基变量的检验数< 0

$$\bar{c}_2 = 29 - [4 \quad 0 \quad 10] \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = 3$$

$$\bar{c}_4 = 38 - [4 \quad 0 \quad 10] \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = -2$$

$$\bar{c}_4 = 0 - [4 \quad 0 \quad 10] \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = 10$$

选择x_4，影子价格小于0，有缩小目标的可能。

step 4： 寻找可改进方向

$$Bd = \begin{bmatrix} 1 & 5 & 0 \\ 0 & 1 & 1 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_3 \\ d_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = A_j$$

这里的A_j是进基变量那列的系数.
求得d:

$$\begin{bmatrix} d_1 \\ d_3 \\ d_5 \end{bmatrix} = \begin{bmatrix} 20/11 \\ -4/11 \\ 4/11 \end{bmatrix}$$

step 5：选择出基变量
希望基变量沿着 d 方向走 θ 步长 (默认$\theta \geq 0$)，仍然保持可行性

$x_{B_i} - \theta d_{B_i} \geq 0$

$$\bar{x}_B - \theta d = \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_3 \\ d_5 \end{bmatrix} \begin{bmatrix} 20/11 \\ -4/11 \\ 4/11 \end{bmatrix}$$

如果出现所有的d_{B_i}都非正，那么x沿着该方向，θ可以无限增大，则问题无界。

选 $max\{\theta\}$, 此时选x_1, x_5都可，不妨选x_1

step6 更新 B 和 x_B

$$B=\begin{bmatrix}0 & 5 & 0\\ 0 & 1 & 1\\ 4 & -1 & 0\end{bmatrix}$$

$$\bar{x}_B=\begin{bmatrix}5.5\\ 12\\ 0\end{bmatrix}$$

重复上述步骤,经检验数检验，此时无进基变量，已求得最优

1.2 按列建模

在使用cplex实现列生成时，需要按列建模
已经熟悉的读者，可自行跳过

```
//Ax=b
//写入空约束框架
IloRange[] rng=new IloRange[mRow];//m行
           for(int i=0;i<mRow;i++)
           {
           //添加约束左右边界
               rng[i]=master.addRange(lb[i], ub[i]);
           }
  IloNumVar[] X=new IloNumVar[nCol];//n列
         for(int j=0;j<nCol;j++)
         {
             IloColumn column=model.column(Obj,c[j]);
             for(int i=0;i<mRow;i++)
             {
                column=column.and(model.column(rng[i],A[i][j]));
             }

X[j]=model.numVar(column,0.0,Double.MAX_VALUE,IloNumVarType.Float);//IloNumVarTyp
e.Float可缺省,
          }
```

2 列生成

从上面的单纯形法，可以学到:一开始使用不完整模型（仅用基变量，而非基变量最后为0），然后利用检验数，不断替换新列，就是在进行列生成。

从下料问题引入

示例2[2]

需求尺寸	需求数量
2.0	48
4.5	35
5.0	24
5.5	10
7.5	8

原料长度设定为 11

step 1: 基与基变量

不妨构造单切方案，每根木料只用来做一种成品
得到受限主问题RMP

$$
\begin{aligned}
min \quad & z = x_1 + x_2 + x_3 + x_4 + x_5 \\
s.t. \quad & 5x_1 \geq 48 \\
& 2x_2 \geq 35 \\
& 2x_3 \geq 24 \\
& 2x_4 \geq 10 \\
& x_5 \geq 8
\end{aligned}
$$

其中5=11/2整除,其他系数同理

step 2: 求解RMP，同时获得对偶值

step 3: 利用检验数，生成新列

$\bar{c}_j = c_j - \pi A_j$

由于方案（可能）无法穷举，因此 此时A_j当作变量求解，所以生成的列实际上是约束矩阵的一列系数，（且用于进基的那列）

step 4: 由于使用求解器，非基变量保留在模型中，不再换出。
step 5: 重复求解，直到达到两个停止条件之一，停止
给出完整的代码

```
public class ColumnGeneration {
    /*
     * RMP
     * min \sum_i{1-5} x_i
     * 5x1⩾48  //[11/2]=5  切成2m长的木品5个
     * 2x2⩾35
     * 2x3⩾24
     * 2x4⩾10
     * x5⩾8
     * x_i⩾0
     */
    static double[] demand_amount= {48,35,24,10,8};//每种木材需求数量
    static double[] demand_size= {2.0,4.5,5.0,5.5,7.5};//每种成品尺寸
    static double W=11;//原料长度
    static int nRow=demand_amount.length;//约束数
    static int nCol=demand_size.length;//变量数  5种单切方案

    IloCplex master;
    IloRange[] rng;
    double eps=1.0e-6;//设定
     IloNumVarArray Cut;
     IloObjective masterObj;
    //按列建模—RMP
    public void buildModelByColumn() throws IloException
    {
        master=new IloCplex();
        master.setParam(IloCplex.Param.RootAlgorithm,IloCplex.Algorithm.Primal);
        masterObj=master.addMinimize();//主问题目标
        rng=new IloRange[demand_amount.length];
       for(int i=0;i<demand_amount.length;i++)
       {
           //添加约束上下界
```

```
            rng[i]=master.addRange(demand_amount[i], Double.MAX_VALUE);
        }

         Cut=new IloNumVarArray();//切割方案 X
        for(int j=0;j<nCol;j++)
        {
            IloColumn col=master.column(masterObj,1.0);
            col=col.and(master.column(rng[j],(int)(W/demand_size[j])));
            Cut.add(master.numVar(col,0.0,Double.MAX_VALUE));//变量初始化
        }
    }
    //定价子问题—生成新的列
    /*
     * 检验数 c_j-u^t A_j
     * min 1.0-u^t A_j
     */
    IloCplex sub;
    IloObjective ReducedCost;
    IloNumVar[] NonB;
    public void buildSp() throws IloException
    {
        sub=new IloCplex();
        NonB=sub.numVarArray(nRow, 0.,Double.MAX_VALUE,IloNumVarType.Int);//A_j


        //约束 L_i cut_i≤W 一根木料切出的成品总长不超过原木料的程度
        sub.addRange(-Double.MAX_VALUE,sub.scalProd(demand_size,NonB),W);
    }
    public void solve() throws IloException
    {
        buildModelByColumn();

        buildSp();
        for(;;)
        {
            master.solve();
            double[] shadowPrice=master.getDuals(rng);

            //更新目标子问题目标函数(或者说更新检验数)
             //ReducedCost.setExpr(sub.diff(1.0, sub.scalProd(NonB,
shadowPrice)));
                if(ReducedCost!=null)
                {
                    sub.delete(ReducedCost);
                    ReducedCost=null;
                }
                IloNumExpr subObj=sub.diff(1.0, sub.scalProd(NonB, shadowPrice));
                ReducedCost=sub.addMinimize(subObj);

                if(sub.solve())
                {
                    //对于min 检验数≥0停止
                    if(sub.getObjValue()>-eps)
                    {
                        break;
                    }
                    double[] newPatt=sub.getValues(NonB);
                    //向RMP添加新的一列
```

```
                    IloColumn column=master.column(masterObj,1.0);
                    for(int i=0;i<newPatt.length;i++)
                    {
                        column=column.and(master.column(rng[i],newPatt[i]));
                    }
                    Cut.add(master.numVar(column,0.0,Double.MAX_VALUE));
                }
        }
        //转换变量类型：IloNumVarArray--IloNumVar[]
           for ( int i = 0; i < Cut.getSize(); i++ )
           {
               //当补齐主问题的列,可以设定求解变量类型
               master.add(master.conversion(Cut.getElement(i),
                                                IloNumVarType.Int));
            }
           master.solve();

           System.out.println("masterObj="+master.getObjValue());
           for(int i=0;i<Cut.getSize();i++)
           {
               System.out.println("Cut"+i+" =
"+master.getValue(Cut.getElement(i)));
           }
           //master.exportModel("cutPattern.lp");
           master.end();
           sub.end();
    }
     //为防止变量长度不够，设计"自动增长"
       static class IloNumVarArray{
           int _num=0;
           IloNumVar[] _array=new IloNumVar[32];
           void add(IloNumVar ivar)
           {
               if(_num≥_array.length)
               {
                   IloNumVar[] array=new IloNumVar[2*_array.length];
                   System.arraycopy(_array, 0, array, 0, _num);//将旧数组内容复制到
新数组
                   _array=array;
               }
               _array[_num++]=ivar;
           }
           IloNumVar getElement(int i)
           {
               return _array[i];//统计变量值
           }
           int getSize()
           {
               return _num; //统计变量数
           }

       }

    public static void main(String[] args) throws IloException {
        ColumnGeneration cg=new ColumnGeneration();
        cg.solve();
    }
```

注意事项：
（1）下料是整数规划，列生成是针对线性规划求解，是有分数解的；

```
master.add(master.conversion(Cut.getElement(i),
                                                IloNumVarType.Int));
```

当找到所有列，改变了变量的类型，使用整数规划求解

（2）cplex 12.10左右版本保留setExpr()

```
ReducedCost.setExpr(sub.diff(1.0, sub.scalProd(NonB, shadowPrice)));
```

但是20以上版本，会报多目标错误，读者们自行选择使用

参考文献

[1] Winston W L. *Operations Research: Applications and Algorithms*, Thomson Learning, Inc., 2004.

[2] Methods and Models for Combinatorial Optimization Column generation methods