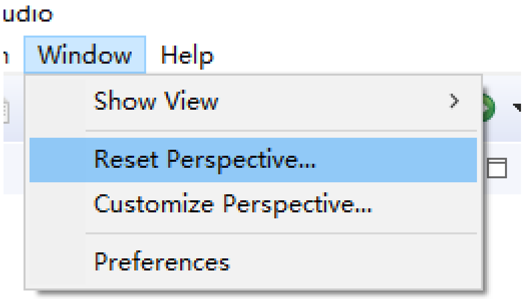
IBM ILOG CPLEX优化软件

陈政培 17363011 智能科学与技术 智科1班

使用IBM ILOG OPL求解规划问题

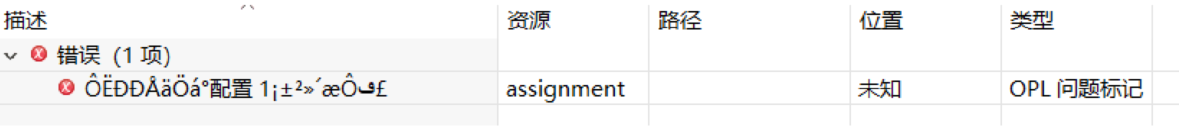
入门

准备工作



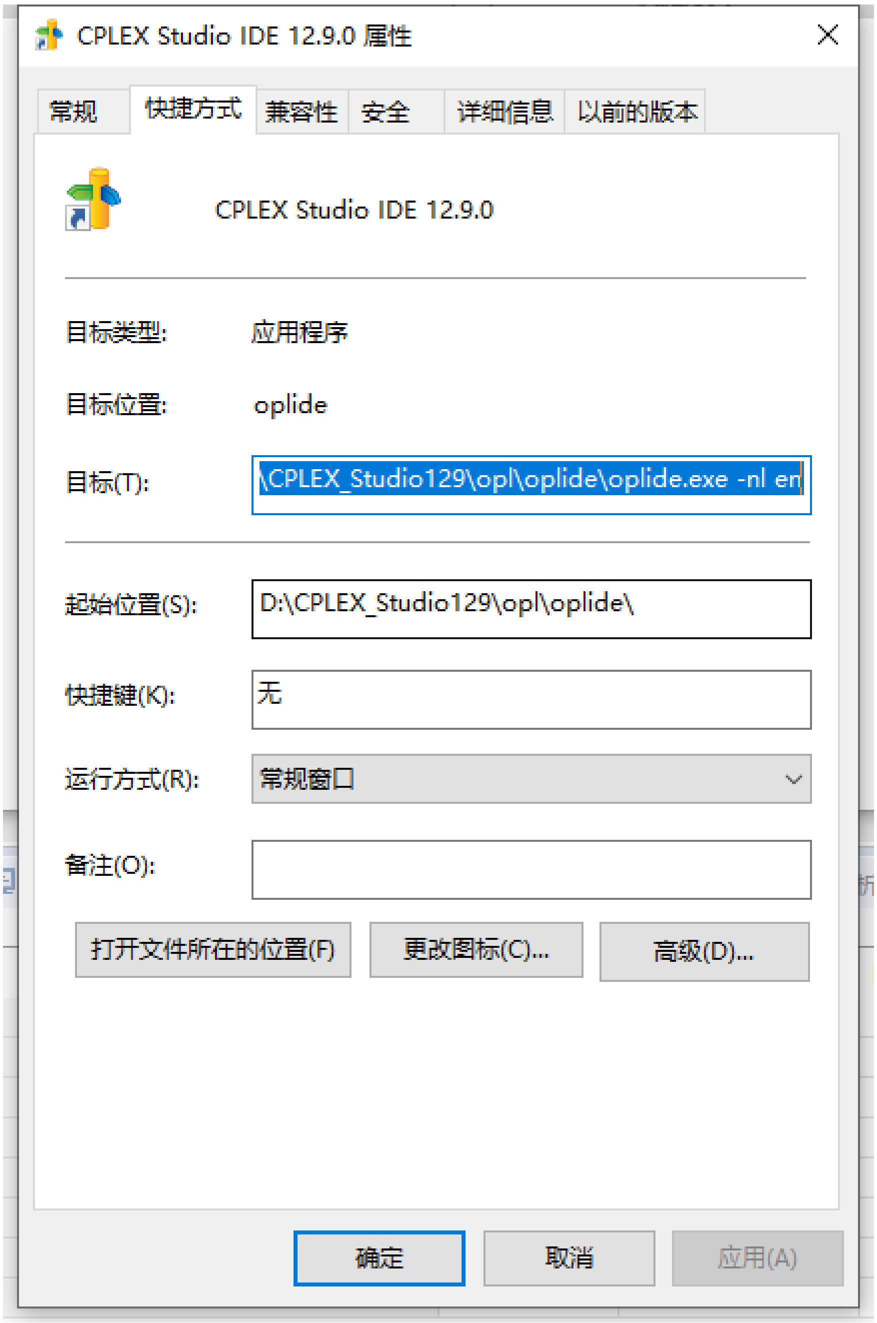
新建项目后，先复位OPL透视图将所有界面恢复到最初的配置

报错



描述	资源	路径	位置	类型
错误（1 项）				
ÔËÐÐÅäÖá°配置 1¡±²»´æÔڡ£	assignment		未知	OPL 问题标记

这个主要是汉化导致的错误，退出程序，对软件启动文件右键属性修改目标属性，加上 `-nl en` 以后软件将以英文启动，并且不会出现中文相关的报错了


CPLEX Studio IDE 12.9.0 属性

CPLEX Studio IDE 12.9.0

目标类型: 应用程序

目标位置: oplide

目标(T): \CPLEX_Studio129\opl\oplide\oplide.exe -nl en

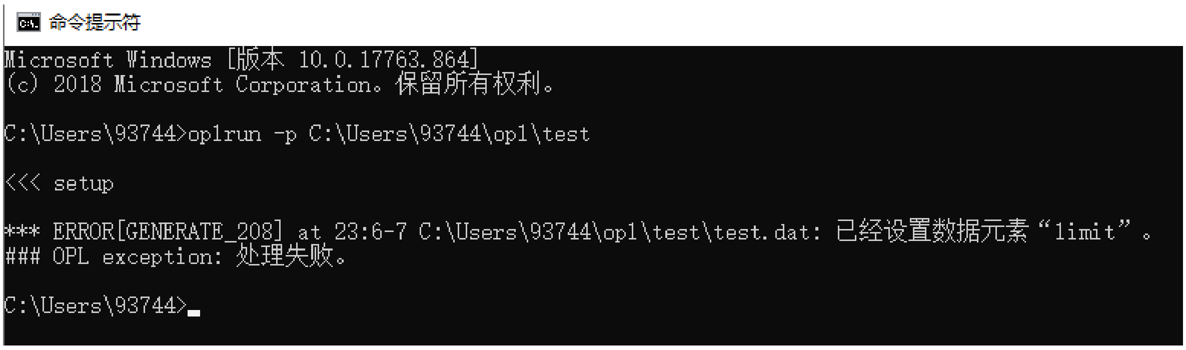
起始位置(S): D:\CPLEX_Studio129\opl\oplide\

快捷键(K): 无

运行方式(R): 常规窗口

备注(O):

无法正常显示报错信息

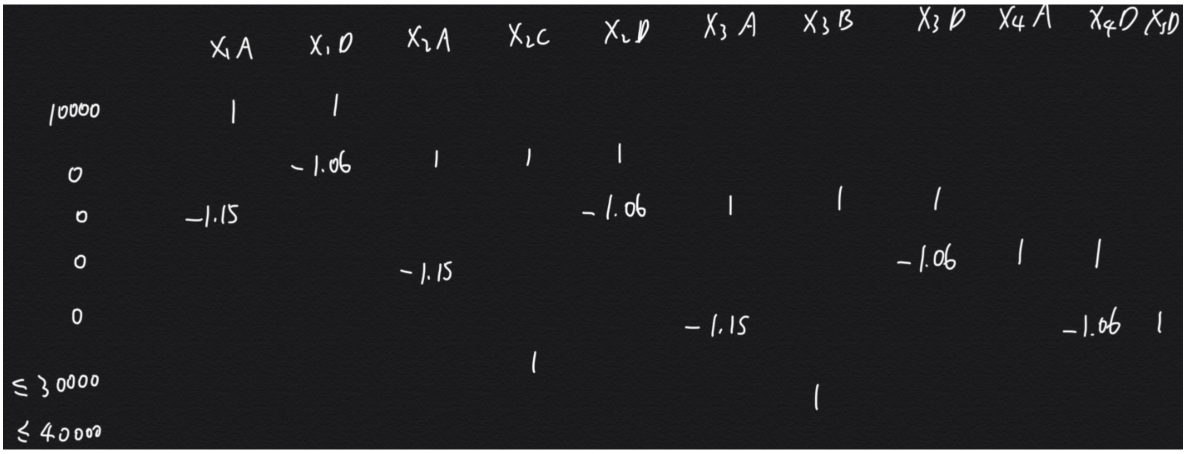


```
Microsoft Windows [版本 10.0.17763.864]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\93744>oplrun -p C:\Users\93744\opl\test

<<< setup

*** ERROR[GENERATE_208] at 23:6-7 C:\Users\93744\opl\test\test.dat: 已经设置数据元素“limit”。
### OPL exception: 处理失败。

C:\Users\93744>
```

在cmd中通过 `oplrun -p + 地址` 得到正常显示的报错信息

第一题——例2-13连续投资问题

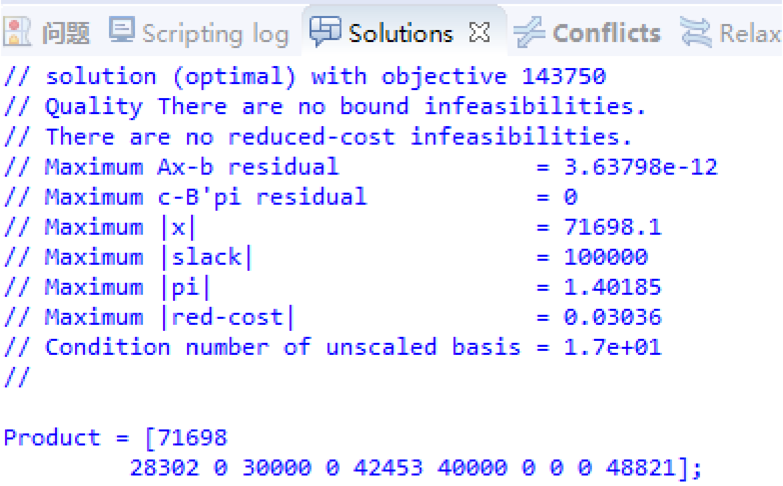
抽象出结构



	X_1A	X_1D	X_2A	X_2C	X_2D	X_3A	X_3B	X_3D	X_4A	X_4D	X_5D
10000	1	1									
0		−1.06	1	1	1						
0	−1.15				−1.06	1	1	1			
0			−1.15					−1.06	1	1	
0						−1.15				−1.06	1
≤ 30000				1							
≤ 40000							1				

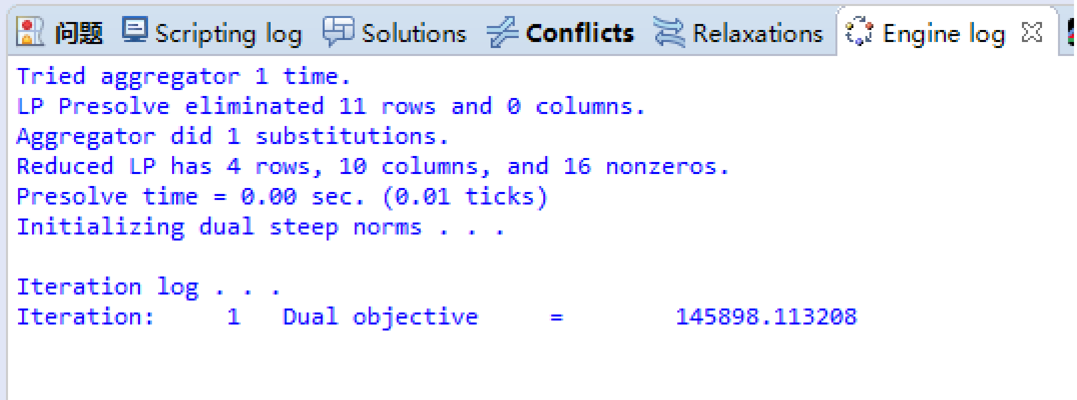
分离数据，完成代码，debug后运行

运行结果（最优解和最优目标）



```
// solution (optimal) with objective 143750
// Quality There are no bound infeasibilities.
// There are no reduced-cost infeasibilities.
// Maximum Ax-b residual              = 3.63798e-12
// Maximum c-B'pi residual            = 0
// Maximum |x|                        = 71698.1
// Maximum |slack|                    = 100000
// Maximum |pi|                       = 1.40185
// Maximum |red-cost|                 = 0.03036
// Condition number of unscaled basis = 1.7e+01
//

Product = [71698
         28302 0 30000 0 42453 40000 0 0 0 48821];
```

Engine Log记录



```
Tried aggregator 1 time.
LP Presolve eliminated 11 rows and 0 columns.
Aggregator did 1 substitutions.
Reduced LP has 4 rows, 10 columns, and 16 nonzeros.
Presolve time = 0.00 sec. (0.01 ticks)
Initializing dual steep norms . . .

Iteration log . . .
Iteration:     1   Dual objective     =        145898.113208
```

第二题——习题2-11 工厂生产问题

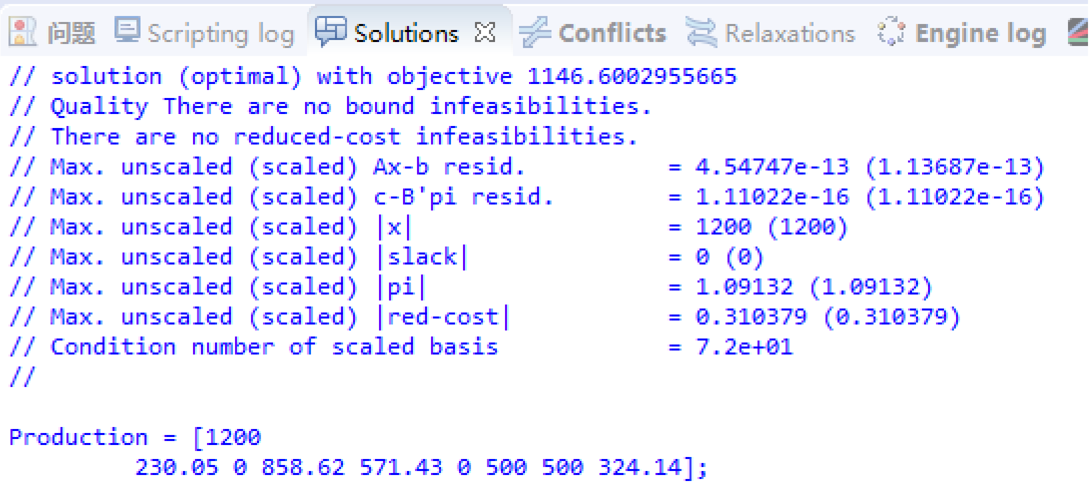
数学模型

$$\begin{aligned}\max z=&(1.25-0.25)\times(x_1+x_2)+(2.00-0.35)\times(x_6+x_7)\\&+(2.80-0.50)\times x_9-\frac{300}{6\,000}\times(5x_1+10x_6)\\&-\frac{321}{10\,000}\times(7x_2+9x_7+12x_9)-\frac{250}{4\,000}\times(6x_3+8x_8)\\&-\frac{783}{7\,000}\times(4x_4+11x_9)-\frac{200}{4\,000}\times 7x_5\end{aligned}$$

$$\text{s. t.}\begin{cases}5x_1+10x_6\leqslant 6\,000\\7x_2+9x_7+12x_9\leqslant 10\,000\\6x_3+\ 8x_8\leqslant 4\,000\\4x_4+11x_9\leqslant 7\,000\\7x_5\leqslant 4\,000\\x_1+x_2=x_3+x_4+x_5\\x_6+x_7=x_8\\x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9\geqslant 0\end{cases}$$

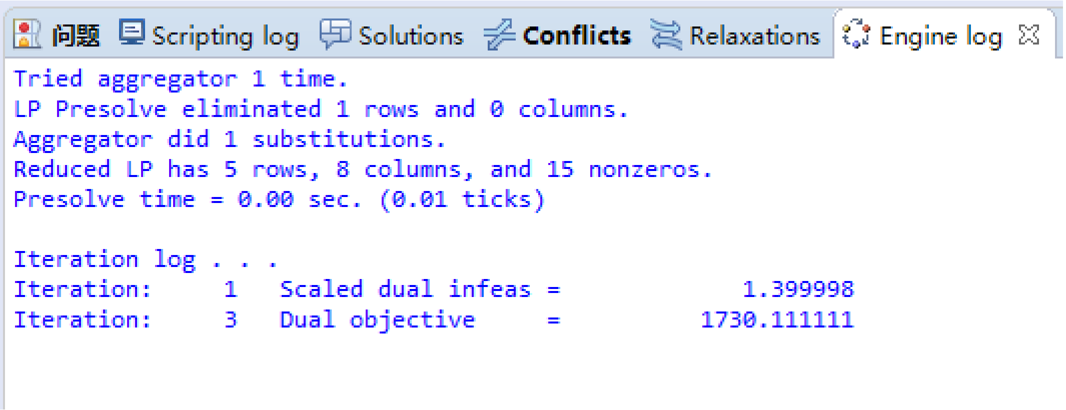
分离数据，完成代码，debug后运行

运行结果（最优解和最优目标）



```
// solution (optimal) with objective 1146.6002955665
// Quality There are no bound infeasibilities.
// There are no reduced-cost infeasibilities.
// Max. unscaled (scaled) Ax-b resid.          = 4.54747e-13 (1.13687e-13)
// Max. unscaled (scaled) c-B'pi resid.        = 1.11022e-16 (1.11022e-16)
// Max. unscaled (scaled) |x|                  = 1200 (1200)
// Max. unscaled (scaled) |slack|              = 0 (0)
// Max. unscaled (scaled) |pi|                 = 1.09132 (1.09132)
// Max. unscaled (scaled) |red-cost|           = 0.310379 (0.310379)
// Condition number of scaled basis            = 7.2e+01
//

Production = [1200
         230.05 0 858.62 571.43 0 500 500 324.14];
```

Engine Log记录



```
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 0 columns.
Aggregator did 1 substitutions.
Reduced LP has 5 rows, 8 columns, and 15 nonzeros.
Presolve time = 0.00 sec. (0.01 ticks)

Iteration log . . .
Iteration:     1   Scaled dual infeas =             1.399998
Iteration:     3   Dual objective     =          1730.111111
```

第三题——统一花费的集合覆盖问题

构建思路

这个问题就类似于`dancing line`问题，难点在于如何生成正确的矩阵

构建输入矩阵

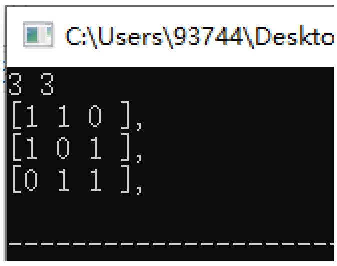
通过递归求出`k`元组和`k+1`元组的所有元素，并保存在两个数组中，并逐一比对是否符合配对关系

```
#include<iostream>
using namespace std;
int main()
{
    int k=5;     //题目要求的k值
    int a=1,b=1;
    for(int i=1;i<=k;i++){
        a=a*(k*2+2-i)/i;
    }//k元组个数
    for(int i=1;i<=k+1;i++){
        b=a*(k*2+2-i)/i;
    }//k+1元组个数
    cout<<a<<" "<<b<<endl;//输出矩阵行和列
    int bi1[a][k],bi2[b][k+1],count=0,t[k+2],flag=0;
    for(int i=0;i<k+2;i++){
        t[i]=i;
    }//t数组辅助递归的实现，先初始化
    while(t[1]<=k+2){
        for(int i=0;i<k;i++){
            bi1[count][i]=t[i+1];
        }//记录当前递归的情况
        ++t[k];
        if(t[k]>k*2+1){
            flag=1;
            while(flag!=0 && flag<k){
                t[k-flag+1]--;
                t[k-flag]++;
                if(t[k-flag]<t[k-flag+1]){
                    while(flag!=0){
                        t[k-flag+1]=t[k-flag]+1;
                        flag--;
                    }
                }
                else
                    flag++;
            }//递归实现
        }
        count++;
    }
    /*for(int i=0;i<a;i++)
    {
        for(int j=0;j<k;j++)
        {
            cout<<bi1[i][j]<<" ";
        }
        cout<<endl;
    }*/          //输出k元组，用于debug
    cout<<endl;
    count=0;flag=-1;
    for(int i=0;i<k+2;i++){
        t[i]=i;
    }
    while(t[1]<=k+1){    //递归原理相同
```

```
        for(int i=0;i<k+1;i++){
            bi2[count][i]=t[i+1];
        }
        ++t[k+1];
        if(t[k+1]>k*2+1){
            flag=0;
            while(flag!=-1 && flag<k){
                t[k-flag+1]--;
                t[k-flag]++;
                if(t[k-flag]<t[k-flag+1]){
                    while(flag!=-1){
                        t[k-flag+1]=t[k-flag]+1;
                        flag--;
                    }
                }
                else
                    flag++;
            }
        }
        count++;
    }
    /*for(int i=0;i<b;i++){
        for(int j=0;j<k+1;j++)
        {
            cout<<bi2[i][j]<<" ";
        }
        cout<<endl;
    }*/         //输出k+1元组，用于debug
    cout<<endl;
    int axis[a][b],temp;
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            axis[i][j]=0;
        }
    }
    for(int i=0;i<a;i++){           //枚举k元组比对是否符合约定条件
        int z;
        temp=0;
        flag=0;
        for(int j=0;j<b;j++){       //枚举k+1元组
            for(z=0;z<k+1;z++){     //逐个元素比对
                if(bi2[j][z] == bi1[i][temp]){
                    temp++;
                    if(temp==k)
                        break;
                }
            }
            if(temp==k){            //满足约定条件，将矩阵对应元素设置为1
                flag=1;
                axis[i][j]=1;
            }
            temp=0;
        }
    }
    for(int i=0;i<a;i++){           //按照格式输出矩阵
        cout<<"[";
        for(int j=0;j<b;j++){
            cout<<axis[i][j]<<" ";
        }
        cout<<"],"<<endl;
    }
}
```

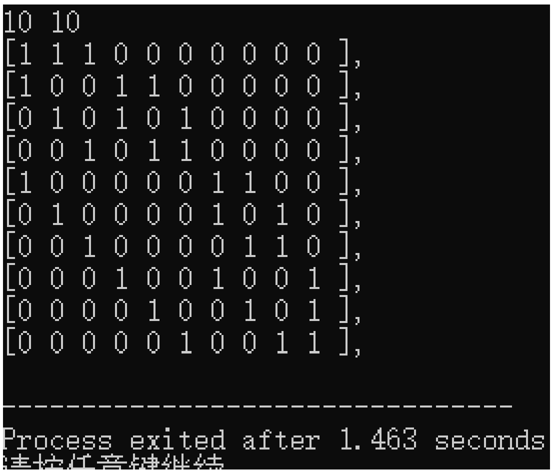
输出样例

- k=1



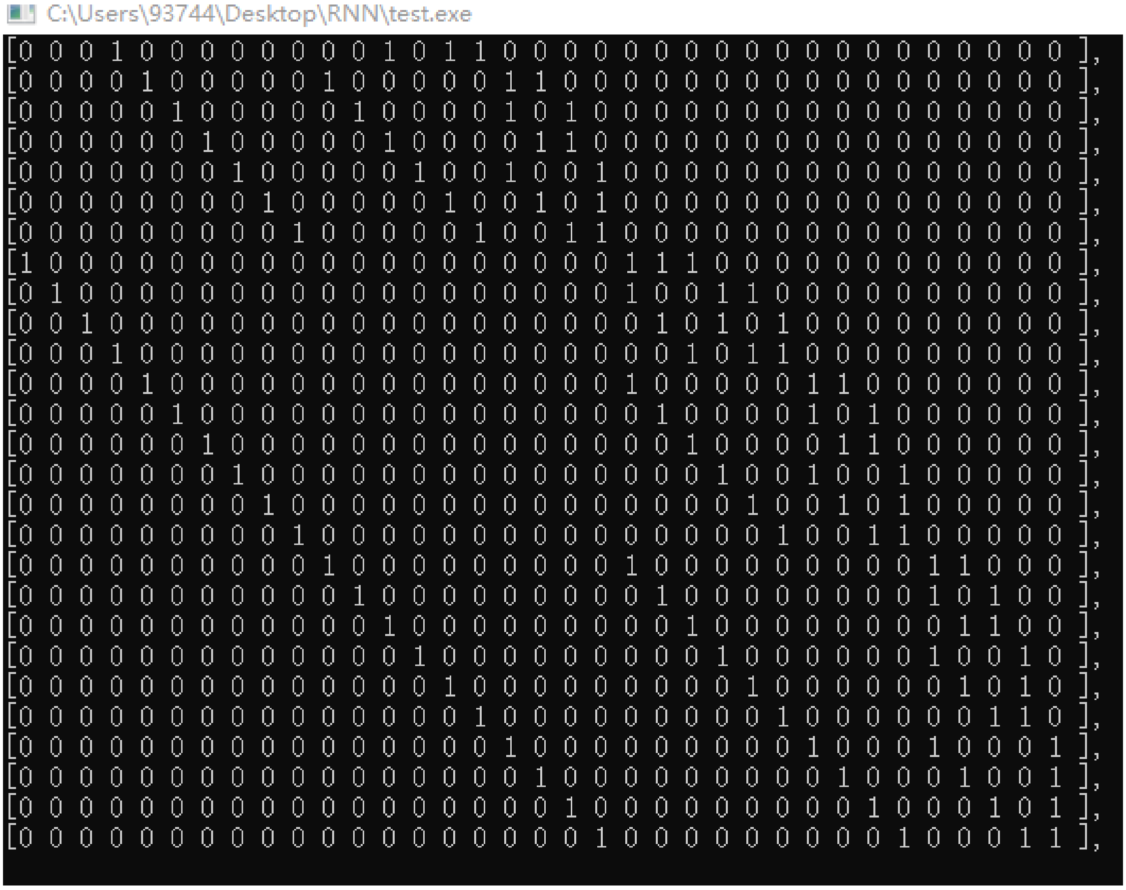
```
3 3
[1 1 0 ],
[1 0 1 ],
[0 1 1 ],

--------------------
```

- k=2



```
10 10
[1 1 1 0 0 0 0 0 0 0 ],
[1 0 0 1 1 0 0 0 0 0 ],
[0 1 0 1 0 1 0 0 0 0 ],
[0 0 1 0 1 1 0 0 0 0 ],
[1 0 0 0 0 0 1 1 0 0 ],
[0 1 0 0 0 0 1 0 1 0 ],
[0 0 1 0 0 0 0 1 1 0 ],
[0 0 0 1 0 0 1 0 0 1 ],
[0 0 0 0 1 0 0 1 0 1 ],
[0 0 0 0 0 1 0 0 1 1 ],

--------------------------------
Process exited after 1.463 seconds
请按任意键继续
```

- k=3



```
[0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ],
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ],
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 ],
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 ],
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 ],
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 ],
[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 ],
[0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 ],
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 ],
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 ],
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 ],
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 ],
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 ],
[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 ],
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 ],
```

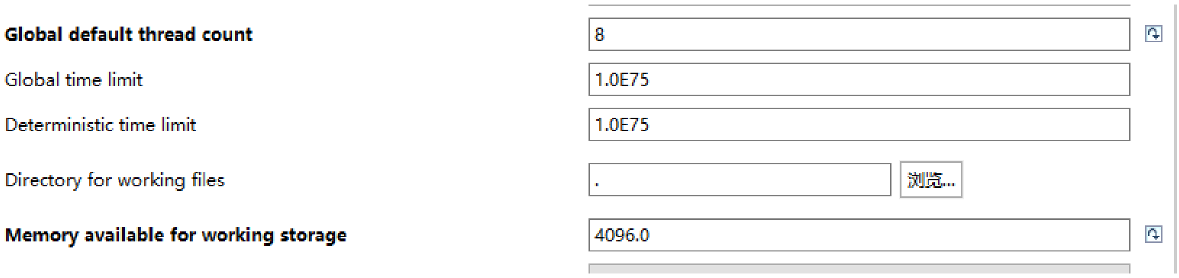
- k=4 k=5矩阵过大，就不截图了

ILOG程序

在解决统一花费的集合覆盖问题时，只需要把抽象的集合问题数学模型中，花费函数 `Cj` 统一定义为1，即每一个`k+1`元组被选取的权重一致，并且此时的 `xj` 为 `bool` 类型变量

OPL ops配置文件设置

在k=1..4时，ILOG都可以很快的计算出问题的最优解，但是当k=5时，数据规模过大，ILOG很长时间无法拟合出最优解，所以需要子在ops中调整一些参数



Global default thread count 8

Global time limit 1.0E75

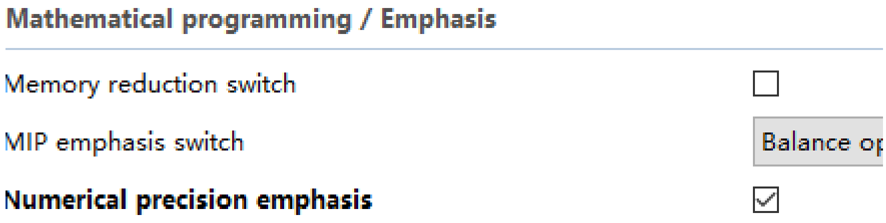
Deterministic time limit 1.0E75

Directory for working files .

Memory available for working storage 4096.0

首先最直观的加速拟合过程的方法就是提高并行计算，将全局线程设置更大，此处考虑到CPU承载能力设置为8线程

随着拟合过程的进行，可行基选取的树结构会越来越复杂，从一开始的几百MB无限增长，所以需要提前设置可用内存，考虑电脑运行内存为8GB，此处设置了4GB



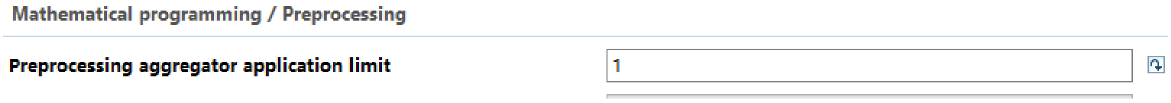
Mathematical programming / Emphasis

Memory reduction switch □

MIP emphasis switch Balance op

Numerical precision emphasis ☑

配置文件默认为平衡效率和稳定性，但是为了更快拟合，需要勾选强调计算精度的选项，可以加快solution拟合



Mathematical programming / Preprocessing

Preprocessing aggregator application limit 1

配置文件中默认为-1自动选择前处理聚合应用程序，此处设置为1对MIP模型变量替换无限制，以改善拟合效率

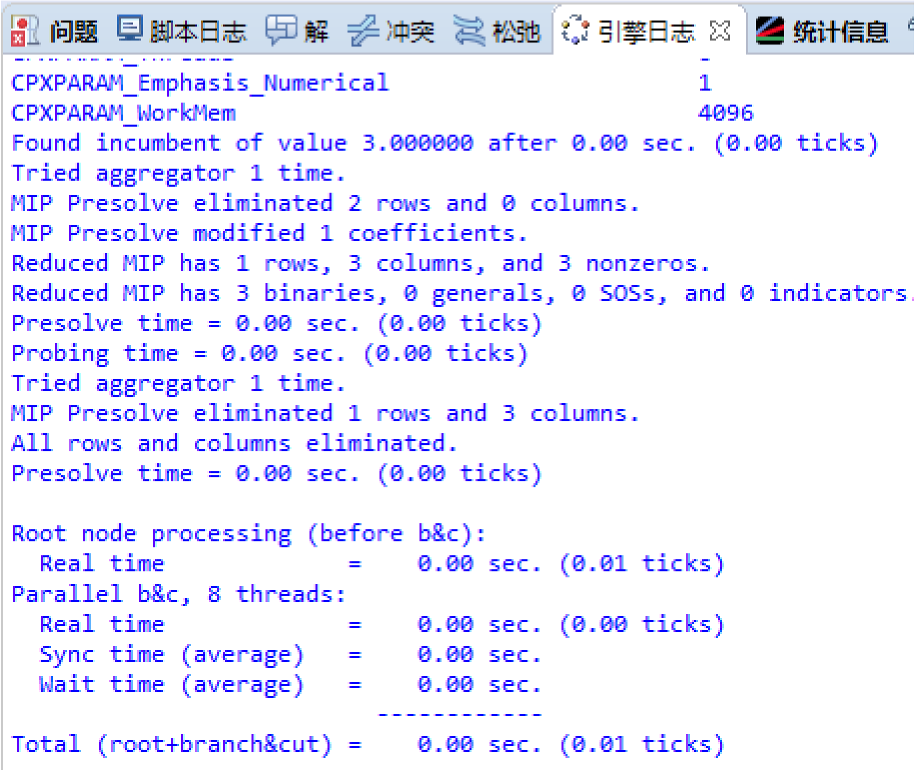
运行结果（最优解、最优目标和Engine Log）（k= 1..4）

- k=1

  最优解和最优目标

```
// solution (optimal) with objective 2
// Quality Incumbent solution:
// MILP objective                                 2.0000000000e+00
// MILP solution norm |x| (Total, Max)            2.00000e+00  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)        0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)                0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)          0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)            0.00000e+00  0.00000e+00
//

x = [1
         1 0];
```

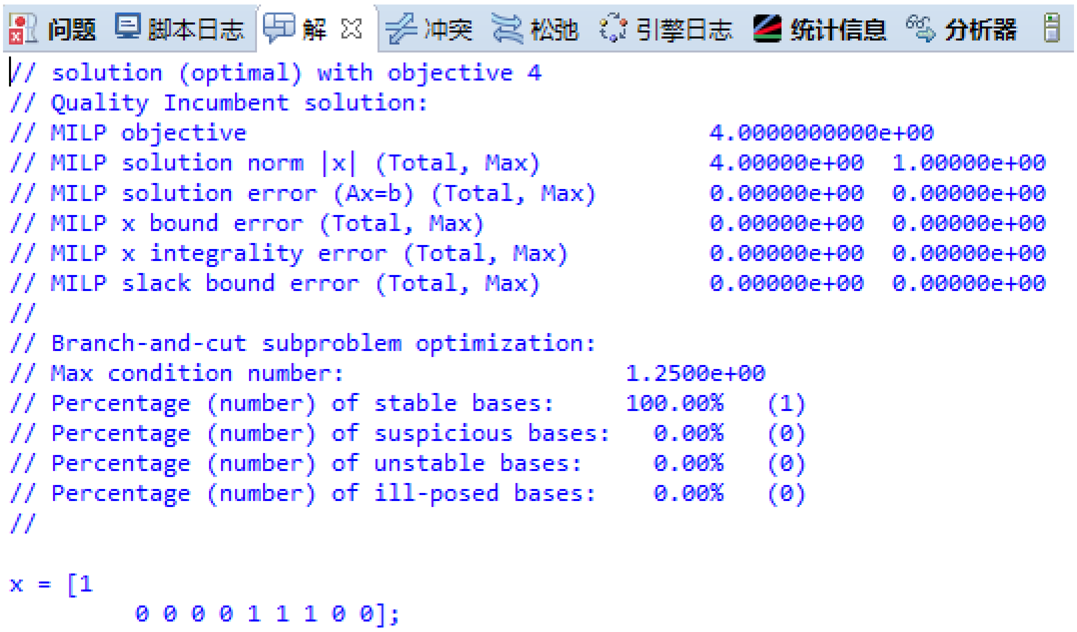
Engine Log



```
CPXPARAM_Emphasis_Numerical                             1
CPXPARAM_WorkMem                                        4096
Found incumbent of value 3.000000 after 0.00 sec. (0.00 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 2 rows and 0 columns.
MIP Presolve modified 1 coefficients.
Reduced MIP has 1 rows, 3 columns, and 3 nonzeros.
Reduced MIP has 3 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.00 ticks)
Probing time = 0.00 sec. (0.00 ticks)
Tried aggregator 1 time.
MIP Presolve eliminated 1 rows and 3 columns.
All rows and columns eliminated.
Presolve time = 0.00 sec. (0.00 ticks)

Root node processing (before b&c):
  Real time             =    0.00 sec. (0.01 ticks)
Parallel b&c, 8 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.00 sec. (0.01 ticks)
```

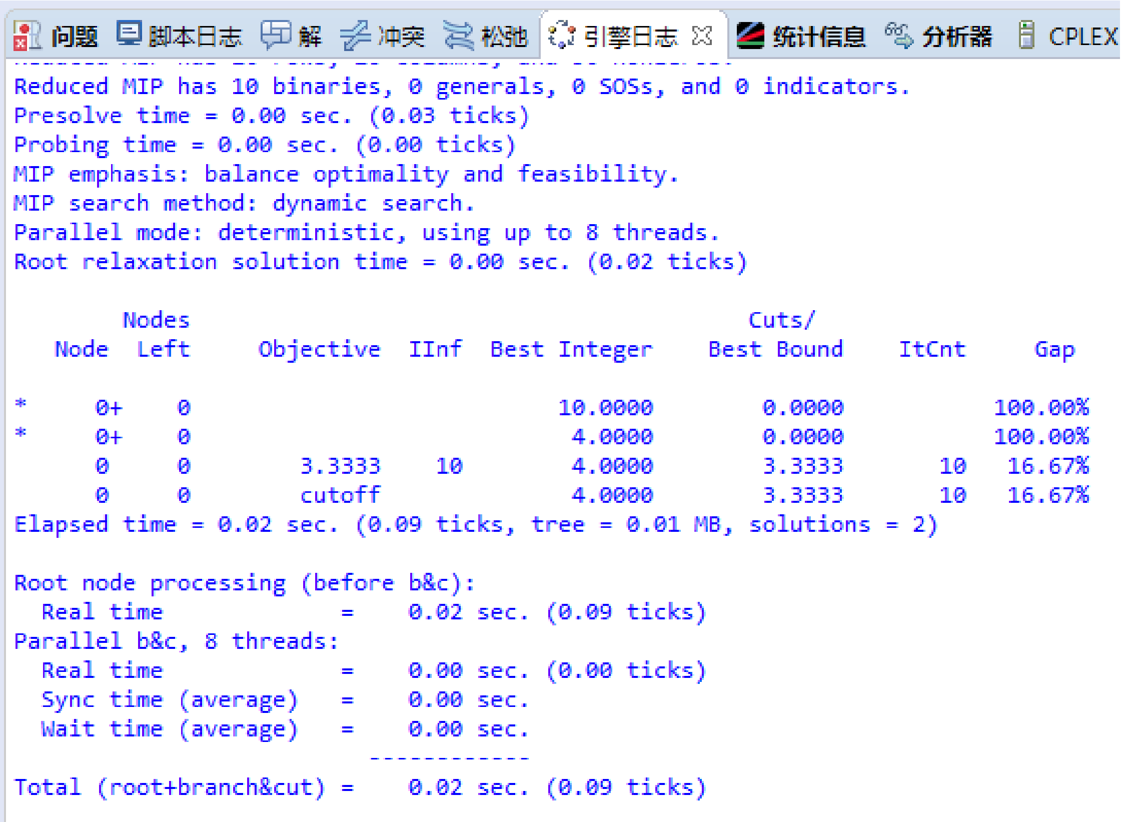
- k=2

最优解和最优目标



```
// solution (optimal) with objective 4
// Quality Incumbent solution:
// MILP objective                                 4.0000000000e+00
// MILP solution norm |x| (Total, Max)            4.00000e+00  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)        0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)                0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)          0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)            0.00000e+00  0.00000e+00
//
// Branch-and-cut subproblem optimization:
// Max condition number:                    1.2500e+00
// Percentage (number) of stable bases:     100.00%   (1)
// Percentage (number) of suspicious bases:   0.00%   (0)
// Percentage (number) of unstable bases:     0.00%   (0)
// Percentage (number) of ill-posed bases:    0.00%   (0)
//

x = [1
         0 0 0 0 1 1 1 0 0];
```

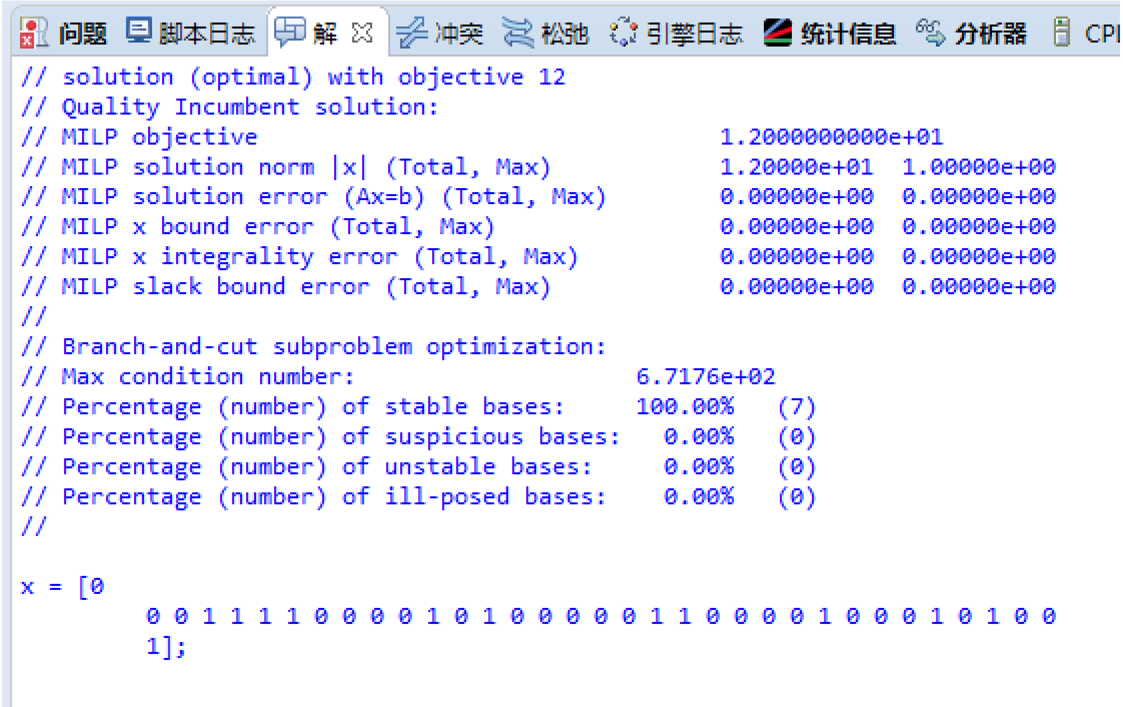
Engine Log



```
Reduced MIP has 10 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.03 ticks)
Probing time = 0.00 sec. (0.00 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.00 sec. (0.02 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                           10.0000        0.0000           100.00%
*     0+    0                            4.0000        0.0000           100.00%
      0     0        3.3333    10        4.0000        3.3333       10   16.67%
      0     0        cutoff              4.0000        3.3333       10   16.67%
Elapsed time = 0.02 sec. (0.09 ticks, tree = 0.01 MB, solutions = 2)

Root node processing (before b&c):
  Real time             =    0.02 sec. (0.09 ticks)
Parallel b&c, 8 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.02 sec. (0.09 ticks)
```

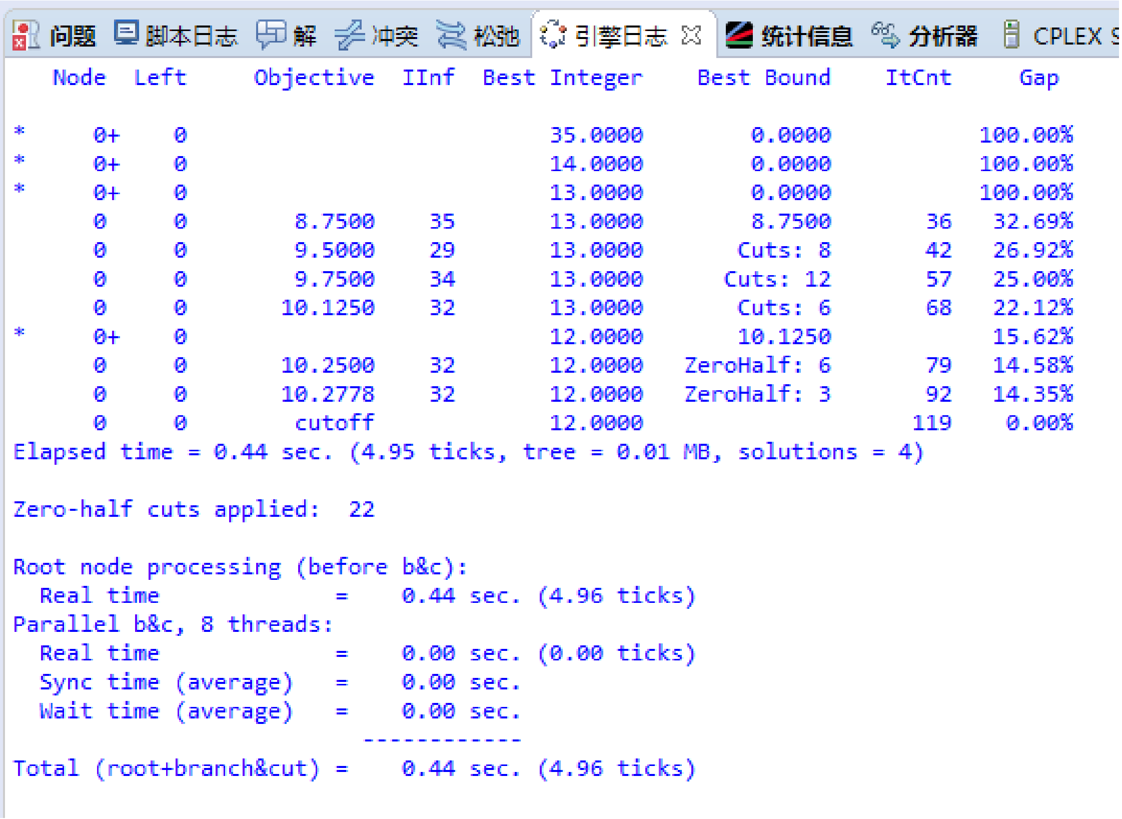
- k=3

  最优解和最优目标



```
// solution (optimal) with objective 12
// Quality Incumbent solution:
// MILP objective                                1.2000000000e+01
// MILP solution norm |x| (Total, Max)           1.20000e+01  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)       0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)               0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)         0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)           0.00000e+00  0.00000e+00
//
// Branch-and-cut subproblem optimization:
// Max condition number:                   6.7176e+02
// Percentage (number) of stable bases:    100.00%   (7)
// Percentage (number) of suspicious bases:  0.00%   (0)
// Percentage (number) of unstable bases:    0.00%   (0)
// Percentage (number) of ill-posed bases:   0.00%   (0)
//

x = [0
         0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0
         1];
```

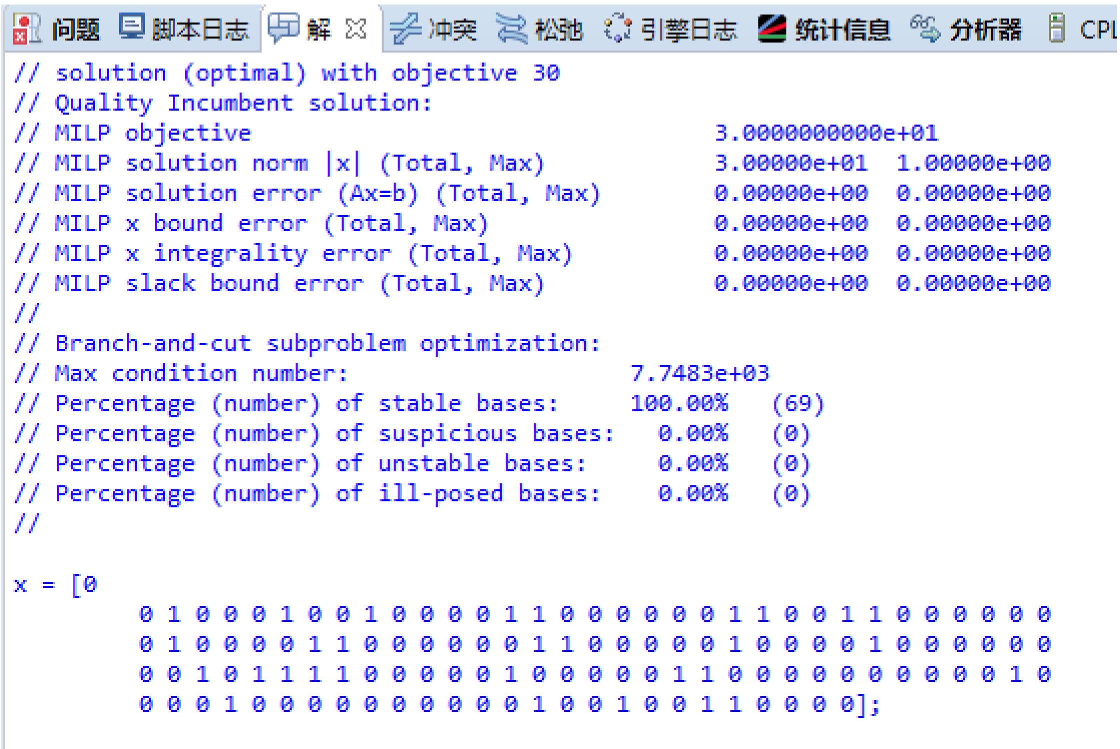
  Engine Log



```
    Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                         35.0000        0.0000           100.00%
*     0+    0                         14.0000        0.0000           100.00%
*     0+    0                         13.0000        0.0000           100.00%
      0     0        8.7500    35     13.0000        8.7500       36   32.69%
      0     0        9.5000    29     13.0000       Cuts: 8       42   26.92%
      0     0        9.7500    34     13.0000      Cuts: 12       57   25.00%
      0     0       10.1250    32     13.0000       Cuts: 6       68   22.12%
*     0+    0                         12.0000       10.1250           15.62%
      0     0       10.2500    32     12.0000    ZeroHalf: 6      79   14.58%
      0     0       10.2778    32     12.0000    ZeroHalf: 3      92   14.35%
      0     0        cutoff           12.0000                    119    0.00%
Elapsed time = 0.44 sec. (4.95 ticks, tree = 0.01 MB, solutions = 4)

Zero-half cuts applied:  22

Root node processing (before b&c):
  Real time             =    0.44 sec. (4.96 ticks)
Parallel b&c, 8 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.44 sec. (4.96 ticks)
```

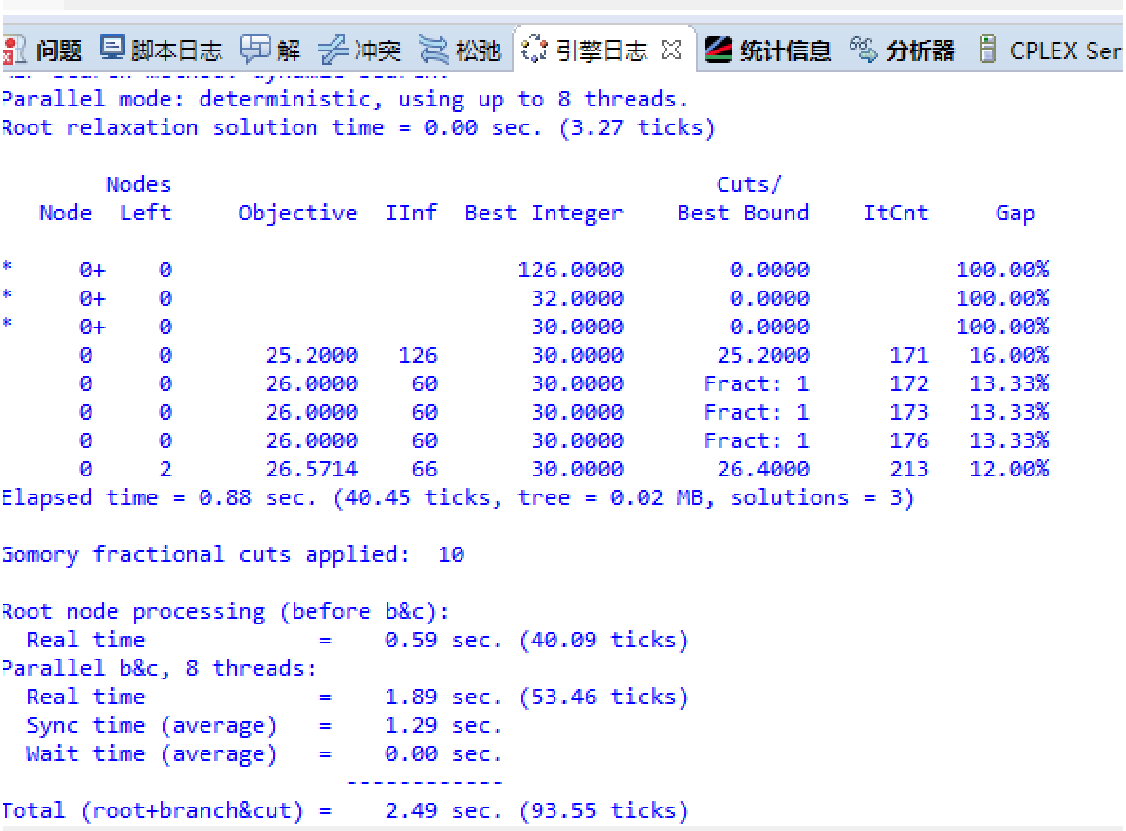
- k=4

最优解和最优目标



```
// solution (optimal) with objective 30
// Quality Incumbent solution:
// MILP objective                                3.0000000000e+01
// MILP solution norm |x| (Total, Max)           3.00000e+01  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)       0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)               0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)         0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)           0.00000e+00  0.00000e+00
//
// Branch-and-cut subproblem optimization:
// Max condition number:                    7.7483e+03
// Percentage (number) of stable bases:     100.00%   (69)
// Percentage (number) of suspicious bases:   0.00%   (0)
// Percentage (number) of unstable bases:     0.00%   (0)
// Percentage (number) of ill-posed bases:    0.00%   (0)
//

x = [0
         0 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0
         0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
         0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0
         0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0];
```

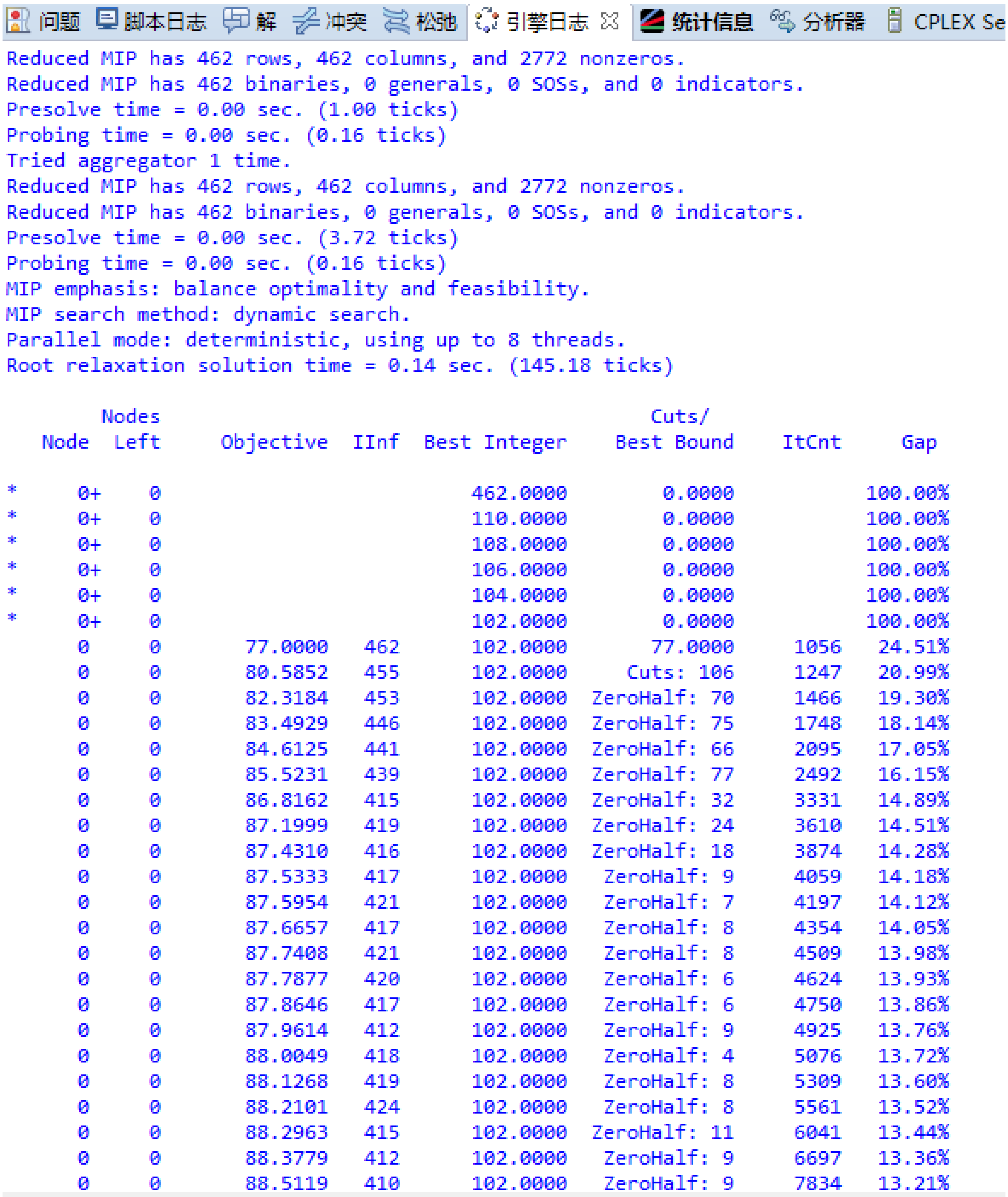
Engine Log



```
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.00 sec. (3.27 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                          126.0000        0.0000           100.00%
*     0+    0                           32.0000        0.0000           100.00%
*     0+    0                           30.0000        0.0000           100.00%
      0     0       25.2000   126       30.0000       25.2000      171   16.00%
      0     0       26.0000    60       30.0000      Fract: 1      172   13.33%
      0     0       26.0000    60       30.0000      Fract: 1      173   13.33%
      0     0       26.0000    60       30.0000      Fract: 1      176   13.33%
      0     2       26.5714    66       30.0000       26.4000      213   12.00%
Elapsed time = 0.88 sec. (40.45 ticks, tree = 0.02 MB, solutions = 3)

Gomory fractional cuts applied:  10

Root node processing (before b&c):
  Real time             =    0.59 sec. (40.09 ticks)
Parallel b&c, 8 threads:
  Real time             =    1.89 sec. (53.46 ticks)
  Sync time (average)   =    1.29 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    2.49 sec. (93.55 ticks)
```

k=5次优解分析

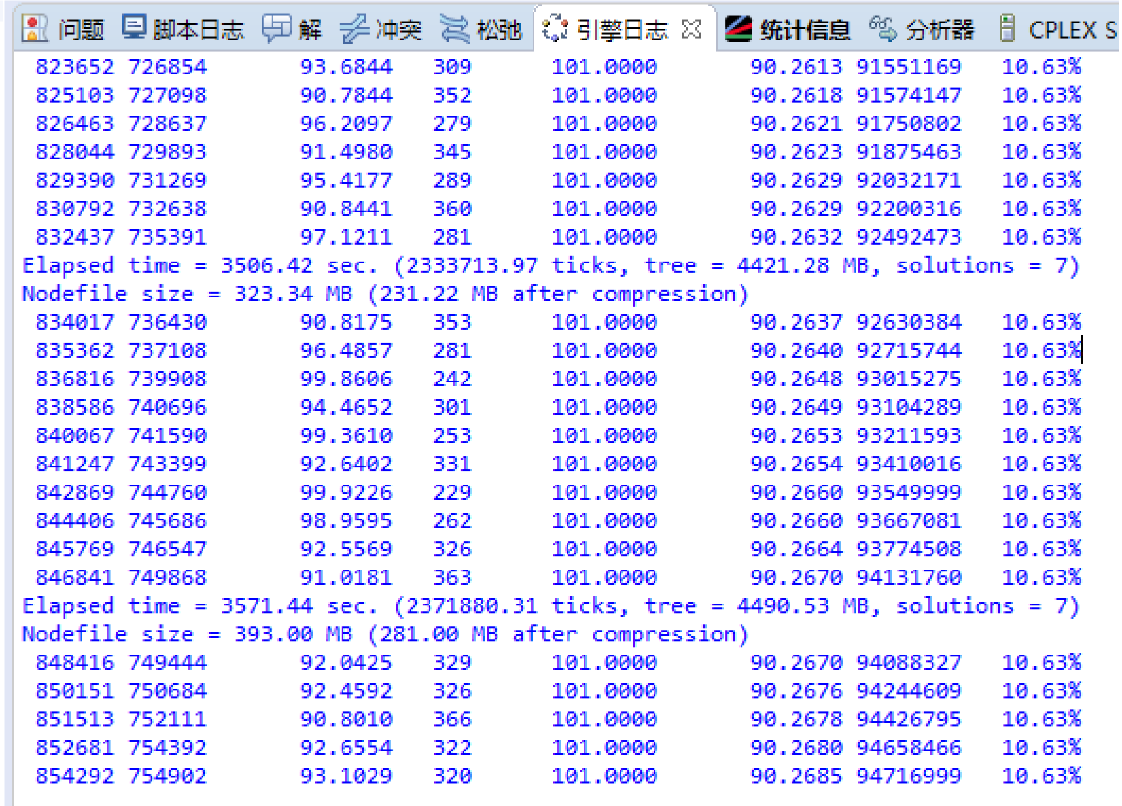
Engine Log 配置好ops后运行



```
Reduced MIP has 462 rows, 462 columns, and 2772 nonzeros.
Reduced MIP has 462 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (1.00 ticks)
Probing time = 0.00 sec. (0.16 ticks)
Tried aggregator 1 time.
Reduced MIP has 462 rows, 462 columns, and 2772 nonzeros.
Reduced MIP has 462 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (3.72 ticks)
Probing time = 0.00 sec. (0.16 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.14 sec. (145.18 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                          462.0000        0.0000           100.00%
*     0+    0                          110.0000        0.0000           100.00%
*     0+    0                          108.0000        0.0000           100.00%
*     0+    0                          106.0000        0.0000           100.00%
*     0+    0                          104.0000        0.0000           100.00%
*     0+    0                          102.0000        0.0000           100.00%
      0     0       77.0000   462      102.0000       77.0000     1056   24.51%
      0     0       80.5852   455      102.0000     Cuts: 106     1247   20.99%
      0     0       82.3184   453      102.0000  ZeroHalf: 70     1466   19.30%
      0     0       83.4929   446      102.0000  ZeroHalf: 75     1748   18.14%
      0     0       84.6125   441      102.0000  ZeroHalf: 66     2095   17.05%
      0     0       85.5231   439      102.0000  ZeroHalf: 77     2492   16.15%
      0     0       86.8162   415      102.0000  ZeroHalf: 32     3331   14.89%
      0     0       87.1999   419      102.0000  ZeroHalf: 24     3610   14.51%
      0     0       87.4310   416      102.0000  ZeroHalf: 18     3874   14.28%
      0     0       87.5333   417      102.0000   ZeroHalf: 9     4059   14.18%
      0     0       87.5954   421      102.0000   ZeroHalf: 7     4197   14.12%
      0     0       87.6657   417      102.0000   ZeroHalf: 8     4354   14.05%
      0     0       87.7408   421      102.0000   ZeroHalf: 8     4509   13.98%
      0     0       87.7877   420      102.0000   ZeroHalf: 6     4624   13.93%
      0     0       87.8646   417      102.0000   ZeroHalf: 6     4750   13.86%
      0     0       87.9614   412      102.0000   ZeroHalf: 9     4925   13.76%
      0     0       88.0049   418      102.0000   ZeroHalf: 4     5076   13.72%
      0     0       88.1268   419      102.0000   ZeroHalf: 8     5309   13.60%
      0     0       88.2101   424      102.0000   ZeroHalf: 8     5561   13.52%
      0     0       88.2963   415      102.0000  ZeroHalf: 11     6041   13.44%
      0     0       88.3779   412      102.0000   ZeroHalf: 9     6697   13.36%
      0     0       88.5119   410      102.0000   ZeroHalf: 9     7834   13.21%
```

Gap收敛速度相对配置前，更快

- 次优解



```
 823652 726854        93.6844   309        101.0000        90.2613 91551169    10.63%
 825103 727098        90.7844   352        101.0000        90.2618 91574147    10.63%
 826463 728637        96.2097   279        101.0000        90.2621 91750802    10.63%
 828044 729893        91.4980   345        101.0000        90.2623 91875463    10.63%
 829390 731269        95.4177   289        101.0000        90.2629 92032171    10.63%
 830792 732638        90.8441   360        101.0000        90.2629 92200316    10.63%
 832437 735391        97.1211   281        101.0000        90.2632 92492473    10.63%
Elapsed time = 3506.42 sec. (2333713.97 ticks, tree = 4421.28 MB, solutions = 7)
Nodefile size = 323.34 MB (231.22 MB after compression)
 834017 736430        90.8175   353        101.0000        90.2637 92630384    10.63%
 835362 737108        96.4857   281        101.0000        90.2640 92715744    10.63%
 836816 739908        99.8606   242        101.0000        90.2648 93015275    10.63%
 838586 740696        94.4652   301        101.0000        90.2649 93104289    10.63%
 840067 741590        99.3610   253        101.0000        90.2653 93211593    10.63%
 841247 743399        92.6402   331        101.0000        90.2654 93410016    10.63%
 842869 744760        99.9226   229        101.0000        90.2660 93549999    10.63%
 844406 745686        98.9595   262        101.0000        90.2660 93667081    10.63%
 845769 746547        92.5569   326        101.0000        90.2664 93774508    10.63%
 846841 749868        91.0181   363        101.0000        90.2670 94131760    10.63%
Elapsed time = 3571.44 sec. (2371880.31 ticks, tree = 4490.53 MB, solutions = 7)
Nodefile size = 393.00 MB (281.00 MB after compression)
 848416 749444        92.0425   329        101.0000        90.2670 94088327    10.63%
 850151 750684        92.4592   326        101.0000        90.2676 94244609    10.63%
 851513 752111        90.8010   366        101.0000        90.2678 94426795    10.63%
 852681 754392        92.6554   322        101.0000        90.2680 94658466    10.63%
 854292 754902        93.1029   320        101.0000        90.2685 94716999    10.63%
```

得到的结果101，后续收敛非常慢，后由于内存溢出停止

以下为log分析，前期收敛较快，从13%到10.64%收敛树结构生成了 `4091.81 MB` ，后开始压缩结构存储，内存报错前 `Gap` 值最优为 `10.63%`

```
          Nodes                                         Cuts/
     Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt
Gap

*     0+    0                         462.0000        0.0000
100.00%
*     0+    0                         110.0000        0.0000
100.00%
*     0+    0                         108.0000        0.0000
100.00%
*     0+    0                         106.0000        0.0000
100.00%
*     0+    0                         104.0000        0.0000
100.00%
*     0+    0                         102.0000        0.0000
100.00%
      0     0       77.0000   462     102.0000       77.0000     1056
24.51%
      0     0       80.5852   455     102.0000      Cuts: 106     1247
20.99%
      0     0       82.3184   453     102.0000   ZeroHalf: 70     1466
19.30%
      0     0       83.4929   446     102.0000   ZeroHalf: 75     1748
18.14%
      0     0       84.6125   441     102.0000   ZeroHalf: 66     2095
17.05%
      0     0       85.5231   439     102.0000   ZeroHalf: 77     2492
16.15%
      0     0       86.8162   415     102.0000   ZeroHalf: 32     3331
14.89%
```

```
      0     0       87.1999   419      102.0000  ZeroHalf: 24     3610
14.51%
      0     0       87.4310   416      102.0000  ZeroHalf: 18     3874
14.28%
      0     0       87.5333   417      102.0000   ZeroHalf: 9     4059
14.18%
      0     0       87.5954   421      102.0000   ZeroHalf: 7     4197
14.12%
      0     0       87.6657   417      102.0000   ZeroHalf: 8     4354
14.05%
      0     0       87.7408   421      102.0000   ZeroHalf: 8     4509
13.98%
      0     0       87.7877   420      102.0000   ZeroHalf: 6     4624
13.93%
      0     0       87.8646   417      102.0000   ZeroHalf: 6     4750
13.86%
      0     0       87.9614   412      102.0000   ZeroHalf: 9     4925
13.76%
      0     0       88.0049   418      102.0000   ZeroHalf: 4     5076
13.72%
      0     0       88.1268   419      102.0000   ZeroHalf: 8     5309
13.60%
      0     0       88.2101   424      102.0000   ZeroHalf: 8     5561
13.52%
      0     0       88.2963   415      102.0000  ZeroHalf: 11     6041
13.44%
      0     0       88.3779   412      102.0000   ZeroHalf: 9     6697
13.36%
      0     0       88.5119   410      102.0000   ZeroHalf: 9     7834
13.21%
      0     0       88.7222   410      102.0000  ZeroHalf: 14     7941
13.02%
      0     0       88.7222   415      102.0000   ZeroHalf: 3     7967
13.02%
      0     2       88.7222   400      102.0000       88.7222     7967
13.02%
...
...
Elapsed time = 392.77 sec. (272300.48 ticks, tree = 384.39 MB, solutions =
7)
  72661  5807       96.1058   294      101.0000       90.0066  9310022
10.88%
Elapsed time = 1151.27 sec. (806900.98 ticks, tree = 1035.82 MB, solutions =
7)
 260167 185744       99.8571   251      101.0000       90.0224 29967702
10.87%
...
...
Elapsed time = 3178.33 sec. (2142853.00 ticks, tree = 3993.34 MB, solutions
= 7)
 773685 678698       93.4005   307      101.0000       90.2499 86090688
10.64%
Elapsed time = 3239.56 sec. (2181024.89 ticks, tree = 4091.81 MB, solutions
= 7)
 788396 692338       91.0870   341      101.0000       90.2540 87645258
10.64%
Elapsed time = 3306.80 sec. (2219194.14 ticks, tree = 4176.81 MB, solutions
= 7)
```

```
Nodefile size = 78.83 MB (56.45 MB after compression)
 802831 705947       93.6489   311      101.0000       90.2569 89199774
10.64%
Elapsed time = 3372.73 sec. (2257370.87 ticks, tree = 4249.38 MB, solutions
= 7)
Nodefile size = 151.58 MB (108.44 MB after compression)
 817920 719181       96.1040   291      101.0000       90.2597 90697048
10.63%
Elapsed time = 3439.75 sec. (2295546.96 ticks, tree = 4327.13 MB, solutions
= 7)
Nodefile size = 228.63 MB (163.49 MB after compression)
 832437 735391       97.1211   281      101.0000       90.2632 92492473
10.63%
Elapsed time = 3506.42 sec. (2333713.97 ticks, tree = 4421.28 MB, solutions
= 7)
Nodefile size = 323.34 MB (231.22 MB after compression)
 846841 749868       91.0181   363      101.0000       90.2670 94131760
10.63%
```

参考文献

1. https://wenku.baidu.com/view/7f9fe7bb48d7c1c709a145dd.html **ILOG_OPL进阶功能**