使用 *lpsolve* 解决线性规划问题

Solve LP problem in lpsolve

一、引言 ***Introduction***

通过一个简单例子来介绍 ***lpsolve*** 求解线性规划问题的方法。

假若农民有 75 亩地，他打算种上两种农作物：小麦和大麦。为了种植这些农作物，农民在种子和化肥等的开销分别为：小麦每亩需要$120，大麦每亩为$210。这个农民可支出的钱有$15000。但是当这些农作物丰收后，需要有仓库来存储，以便在行情最好的时候将这些农作物买出。农民的仓库可存储 4000 蒲式耳(计量谷物等的容量单位，在英国等于 36.368 升，在美国等于 35.238 升)。每亩小麦平均产量为 110 蒲式耳，每亩大麦平均产量为 30 蒲式耳。若每蒲式耳小麦的净利润为$1.30，每蒲式耳大麦的净利润为$2.00，农民怎么样来种植这 75 亩地能获得最大利益？

二、建立数学模型 ***Formulation of an lp problem***

首先，我们求出目标函数（***Objective***），即利润；然后找出约束条件，并画出图形；最后，我们通过图形和一些简单计算得到最优解。

设小麦的种植面积为 x，大麦的种植面积为 y。

则 $P=(110)(1.30)x + (30)(2.00)y = 143x + 60y$ 为目标函数。当其取最大值时也表示农民将获得最大利润。还有如下三个约束不等式，约束分别来自可用支出的限制、存储容量限制和种植面积限制。分别表示如下：

$120x + 210y <= 15000$

$110x + 20y <= 4000$

$x + y <= 75$

严格来说，还有两个不等式的约束，即非负约束来自实际情况，因为农民不可能在负数的地上种植庄稼。即：$x >= 0, y >= 0$。

得数学模型如下所示：

$max(143x + 60y)$

s.t.

$120x + 210y <= 15000$

$110x + 30y <= 4000$

$x + y <= 75$

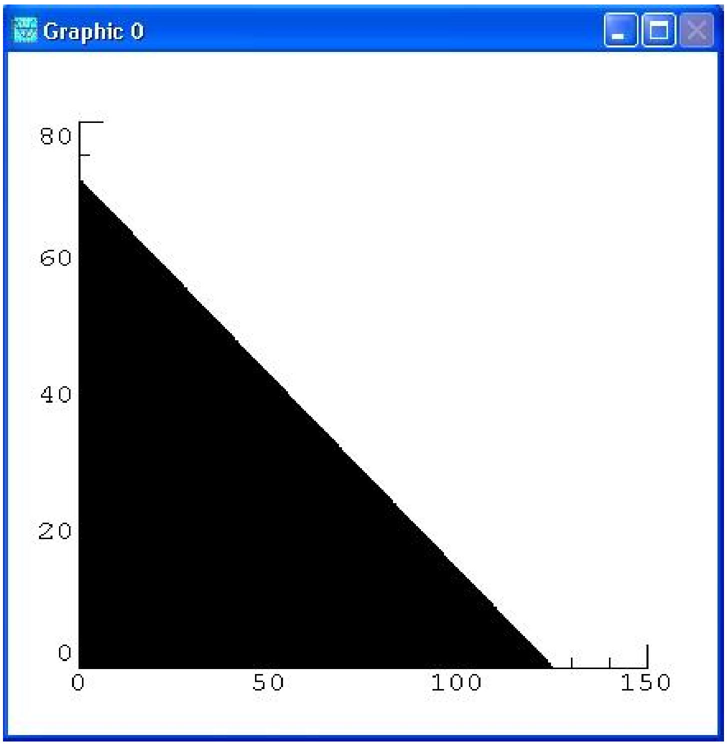
$x >= 0$

$y >= 0$

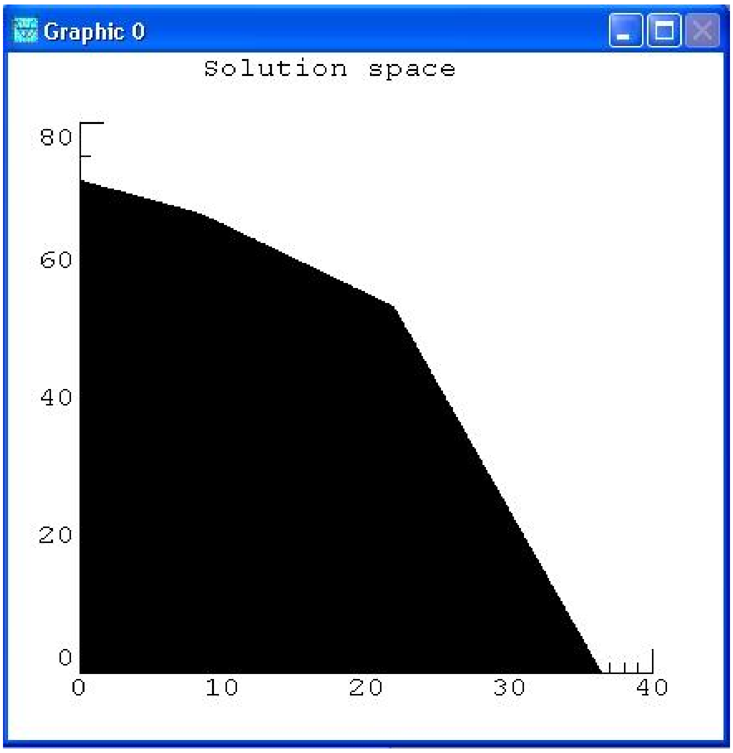
三、图解法

非负约束表明我们只需要考虑 ***X-Y*** 平面中的第一象限即可。$x + y <= 75$ 表示以直线 $x+y=75$ 为界的左下方区域。作图如下所示：

eryar@163.com

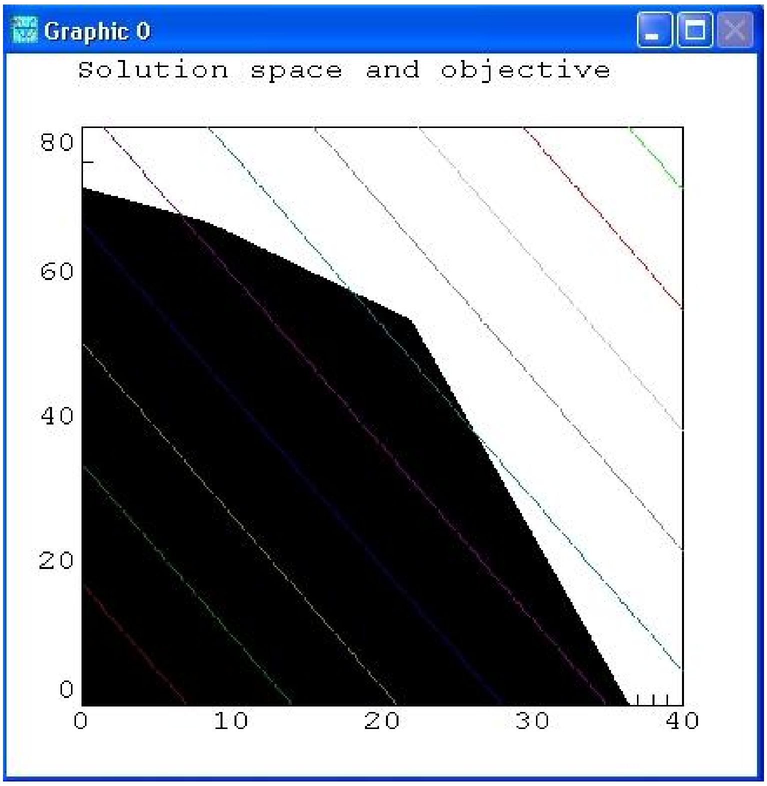


加上另外两个约束条件后，如下图所示：



黑色区域为可行域，即在可行域中任意一点都满足约束条件。

沿目标函数梯度方向作等值线，如下图所示：



等值线与黑色区域相交均为可行解。等值线越向右移动，则目标函数值越大。最优解就是取得最大值且与黑色区域仍有交线的那条等值线。

通过图示，可以清楚地看出目标函数 ***P*** 的最大值在可行域的多边形顶点上取得，坐标值大约为（22, 53）。即直线 ***x***+***y***=75 与直线 110****x***+30****y***=4000 的交点。这个点是可行域多边形的角点。这不是偶然现象，***lpsolve*** 中使用的单纯形算法求得的最优解也是在角点处取得。

定理：对线性规划问题，若可行域有界且存在最优解，则目标函数必可在其可行域的某个顶点达到最优值。

四、使用 ***C***/***C***++建立模型

上述模型在 ***C*** 中建立如下：

```
//------------------------------------------------------------------------------
//    Copyright (c) 2012 eryar All Rights Reserved.
//
//        File     : Main.cpp
//        Author   : eryar@163.com
//        Date     : 2012-9-9 17:11
//        Version  : 0.1v
//
//    Description : lpsolve test program.
//
//==============================================================================
#include <iostream>
using namespace std;

#include "lp_lib.h"
```

```
#pragma comment(lib, "liblpsolve55.lib")

int demo(void);

int main(int argc, char* argv[])
{
    return demo();
}

int demo( void )
{
    lprec*   lp;
    int Ncol        = 0;
    int *colno    = NULL;
    int j              = 0;
    int ret          = 0;
    REAL* row      = NULL;

    // We will build the model row by row,
    // So we start with creating a model with 0 rows and 2 columns.

    // There are two bariables in the model.
    Ncol        = 2;

    lp    = make_lp(0, Ncol);

    if (lp == NULL)
    {
        cout<<"Unable to create new LP model!\n"<<endl;

        ret = 1;
    }

    if (ret == 0)
    {
        // Let us name our bariables. Not required, but can be useful for debugging.
        set_col_name(lp, 1, "x");
        set_col_name(lp, 2, "y");

        // Create space large enough for one row.
        colno      = (int *)malloc(Ncol * sizeof(*colno));
        row          = (REAL*)malloc(Ncol * sizeof(*row));
```

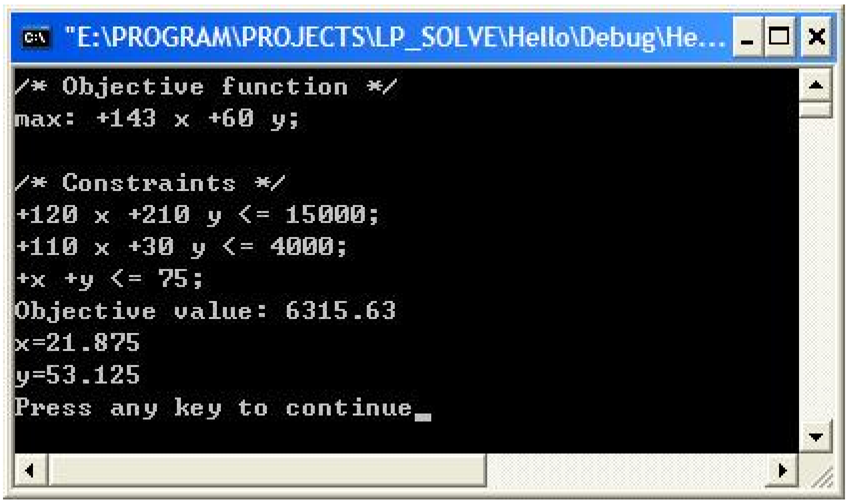
```
        // malloc memory failed.
        if ((colno == NULL) || row == NULL)
        {
            ret = 2;
        }
    }

    if (ret == 0)
    {
        // Makes building the model faster if it is done rows by row.
        set_add_rowmode(lp, TRUE);

        // Construct first row (120 x + 210 y <= 15000).
        j    = 0;

        // First column.
        colno[j]    = 1;
        row[j++]    = 120;

        // Second column.
        colno[j]    = 2;
        row[j++]    = 210;

        // Add the row to lpsolve.
        if (!add_constraintex(lp, j, row, colno, LE, 15000))
        {
            ret = 3;
        }
    }

    // Construct second row (110x + 30y <= 4000).
    if (ret == 0)
    {
        j = 0;

        // First column.
        colno[j]    = 1;
        row[j++]    = 110;

        // Second column.
        colno[j]    = 2;
        row[j++]    = 30;

        // Add the row to lpsolve.
```

```
        if (!add_constraintex(lp, j, row, colno, LE, 4000))
        {
            ret = 3;
        }
    }

    // Construct third row (x + y <= 75).
    if (ret == 0)
    {
        j = 0;

        // First column.
        colno[j]    = 1;
        row[j++]    = 1;

        // Second column.
        colno[j]    = 2;
        row[j++]    = 1;

        // Add the row the lpsolve.
        if (!add_constraintex(lp, j, row, colno, LE, 75))
        {
            ret = 3;
        }
    }

    // Set the objective function (143x + 60y).
    if (ret == 0)
    {
        // Row mode should be truned off again when done building the model.
        set_add_rowmode(lp, FALSE);

        // Set the objective function (143x + 60y).
        j = 0;

        // First column.
        colno[j]    = 1;
        row[j++]    = 143;

        // Second column.
        colno[j]    = 2;
        row[j++]    = 60;

        // Set the objective in lpsolve.
```

```
        if (!set_obj_fnex(lp, j, row, colno))
        {
            ret = 4;
        }
    }

    if (ret == 0)
    {
        // Set the object direction to maximize.
        set_maxim(lp);

        // Just out of curioucity, now show the model in lp format on screen.
        // This only works if this is a console application. If not, use
        // wirte_lp and a file name.
        write_LP(lp, stdout);

        // I only want to see important messages on screen while solving.
        set_verbose(lp, IMPORTANT);

        // Now let lpsolve calculate a solution.
        ret = solve(lp);

        if (ret == OPTIMAL)
        {
            ret = 0;
        }
        else
        {
            ret = 5;
        }
    }

    // A solution is calculated, now lets get some results.
    if (ret == 0)
    {
        // Objective value.
        cout<<"Objective value: "<<get_objective(lp)<<endl;

        // Variable values.
        get_variables(lp, row);

        for (j = 0; j < Ncol; j++)
        {
            cout<<get_col_name(lp, j+1)<<"="<<row[j]<<endl;
```

```
        }

        // We are done now.
    }

    // Free allocated memory.
    if (row != NULL)
    {
        free(row);
    }

    if (colno != NULL)
    {
        free(colno);
    }

    // Clean up such that all used memory by lpsolve is freed.
    if (lp != NULL)
    {
        delete_lp(lp);
    }

    return ret;
}
```

计算结果如下所示：



五、结论

通过这个简单例子，对 **lpsolve** 的使用有了个初步认识。若想进一步，可以参考其文档。