瀋陽工業大學

《数据结构与算法》
课程设计报告

姓　　名	杨二贺	学　　号	180405218
班　　级	计算机科学与技术1802	指 导 教 师	邵虹
实 验 名 称	数 据 结 构 与 算 法		
开 设 学 期	2019-2020 第二学期		
开 设 时 间	第 18 周——第 19 周		
报 告 日 期	2020-7-3		
评 定 成 绩		评　定　人	邵虹
		评 定 日 期	2020-7-9

目　录

1 题目要求

1.1 题目背景

排料在工业生产中具有重要地位，排料时对原材料的利用率直接关系到企业的生产 成本和经济效益。随着计算机辅助工程技术的发展，越来越多的企业使用计算机设计排料方式，从而有效降低生产成本、提高经济效益。矩形优化排料问题是指在给定规格和数量的矩形原材料上尽可能多地排放矩形待切零件，使得各零件紧密排放而不重叠，达到原材料的最大化利用。而且要保证工艺要求、得到较高的原材料空间利用率。

1.2 题目要求

给定若干个零件,规格：长、宽、重量。在给定一个矩形加工区域。需要将这些零件放在给定的矩形加工区域上，求一种或者多种布局方案利用率最优。矩形加工区域规格：长、宽和承载的重量。基于以上条件，在给定零件信息的前提下，生成合理的生产调度方案

2 需求分析

2.1 项目需求概述

2.1.1 项目背景和算法需求

项目背景：项目灵感来自于实际生产作业，在保证多种约束条件的前提下，能够提高对原材料的利用率，从而达到降低成本，提高经济效益的目的。

算法要求：待加工的零件根据多种约束生成队列（每个零件的种类的数目均不一样），放入一个矩形加工区域内，实现在满足约束条件的前提下达到空间利用率最高的情况。

2.1.2 功能介绍

(1)实现系统图形界面
(2)蚁群算法参数输入与控件之间的连接
(3)通过界面交互的方式来读取测试数据文件
(4)坐标系转换,并对结果绘图
(5)约束函数条件计算
(6)更新矩形加工信息

2.1.3 运行环境

（1）操作环境：Windows10
（2）硬件环境：CPU：Intel(R) Cor(TM) i7-8750H CPU @2.20GHz 2.21GHz

（3）内存：8G
（4）软件环境：Qt Creator 4.11.1 (Community)

2.2 功能需求

2.2.1 系统主界面的设计

完成程序运行主界面包括显示读取文件的路径，以及选择文件的按钮，能够弹出选择文件路径的对话框，设计出主界面能够实现调节蚁群算法的参数调节，用户可以通过界面交互形式的修改蚁群算法的信息启发因子，期望启发式因子，全局信息素挥发参数，局部信息素挥发参数, 以及加工区域的宽、高和最大承重等信息。确定按钮能够实现程序的执行，用户的不合法操作给出以相应的警告弹框。

2.2.2 实现结果绘图的对话框

能够实现根据坐标点的绘制加工材料的矩形，要求坐标系需要与现实世界的左下角为原点的坐标系一致。同时不同材料的填充颜色需要不同，并且颜色不能选择过淡的填充色。结果的绘制图形不能过小或者过大，需要对绘制的图形进行合适的放大和缩小调整，以便提高用户观感。结果对话框中也需要包括当前摆放情况的材料对加工区域的利用率。结果用 LCD 样式显示结果。

2.2.3 文件选取对话框的设计

能够通过点击选择文件按钮来实现弹出文件选取的对话框，根据用户的选择操作获取用户选择文件的路径，如果用户没有选取文件，则弹出警告提示框。将获取文件的路径进行临时的存放，用于使用文本流的方式读取用户选择的文件，以便于将数据显示在主界面上。

2.2.4 约束函数的条件计算

根据当前迭代情况，判断放入下一个材料能否放入到空余的区域中，如果能够放入需要保存放置的情况，是否旋转横着放置，以及材料的放置后对应的左下角和右上角坐标。如果不能放置直接返回。

3 概要设计

3.1 系统主界面的设计

首先在设计师界面类里完成基本控件的摆放（图 3-1-1）。根据各个控件摆放的位置，设计出可以合理布局的 Widget，把相应控件放入合适的 Widget 中。对控件设置相应的大小范围，对一些因为加入垂直或水平布局使控件形状发生严重变形的，可以使控件设置为固定大小。然后将不同的 Widget 设置合适的布局方式（图 3-1-2）。最后对总界面设置布局方式，根据结果运行的实际情况在对界面的大小以及各个 Widget 大小做一些合适的调整，达到最后界面的设计（图 3-1-3）。包括显示读取文件的路径信息，以及读取数据的显示。为了减少相应的代码复杂度，控件的部分参数在设计时就完成相应的设置。

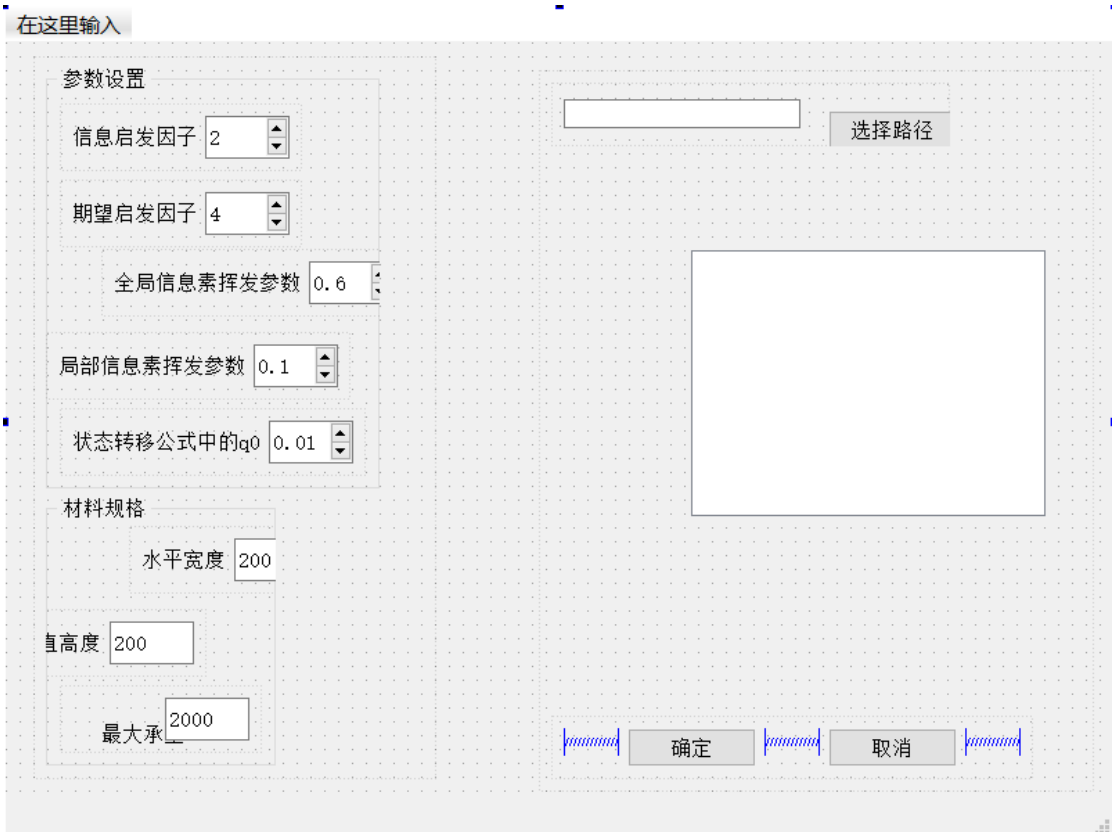


图 3-1-1（基本控件摆放）

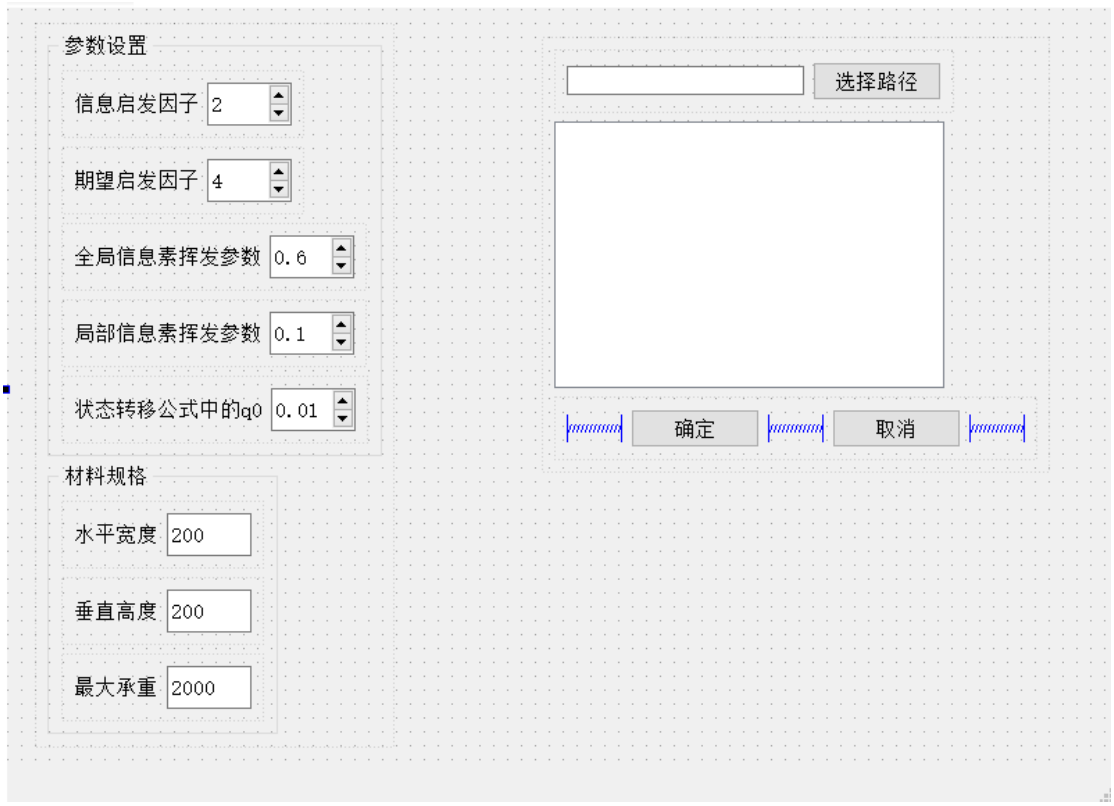


图 3-1-2（加入布局后）

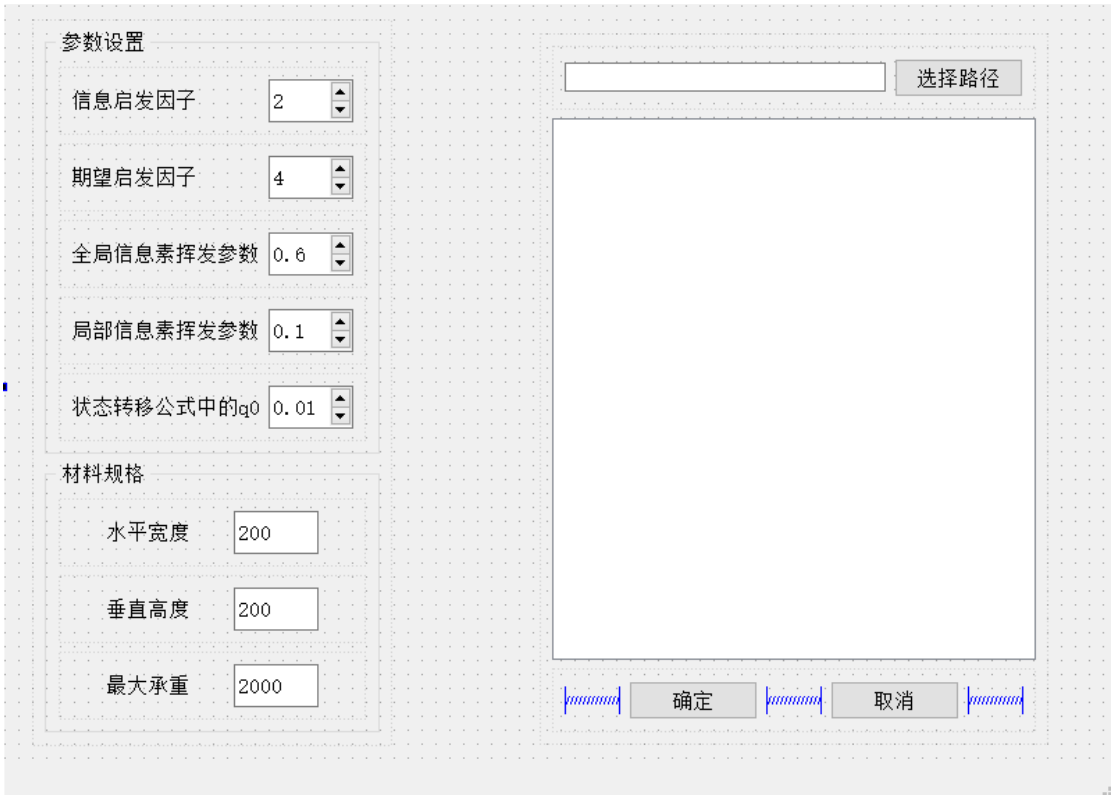


图 3-1-1（最后布局）

3.2 绘图对话框设计

根据结果的范围，在绘图对话框上选择合适的大小用来作为画布来进行绘图，在剩余的区域用来放置显示利用率的 LcdNumber 控件（图 3-2-1），上面放置一个 Lable 标签（图 3-2-2）用于展示 LcdNumber 控件数字代表的含义,设置合适 Lable 的字体类型以及字号。由于绘图界面的坐标系是左上为原点的坐标系，绘制图形需要进行坐标变换，改为左下为原点的坐标系。需要对虚函数 PaintEvent 函数进行 override 用于绘制图形，在程序选择执行的那个确定按钮控件对应的槽函数里面加入 update 函数，用于将绘图结果进行更新，由于 PainEvent 会有 Qt 系统后台默认调用，只需要加入 update 函数就行了。这个界面控件少，总体一个水平布局就可以达到布局的效果。因为程序执行每次只有一种情况，即绘图的绘制结果也只有一种情况，所以需要这个对话框设置成模态对话框，用户必须对这个界面响应才能进行下一次的执行操作。最后界面效果图 3-2-3。

图 3-2-1（LcdNumber 控件）

图 3-2-2（加上 Lable 后）

图 3-2-3（绘图对话框）

3.3 文件读取弹框设计

使用 QFileDialog 类实现文件读取对话框，将获取的字符串保存在 QString 中，并将写在选择路径按钮点击信号对应的槽函数中，然后将其显示在 LineEdit 控件中效果如图 3-3-1，文件对话框显示如图 3-3-2。通过 QFile 将字符串变成地址对象。设置文件的读取方式，通过文本流的方式打开文件，通过字符串的分割，将待测数据保存用于程序执行计算。将选择路径控件的点击信号对应的槽函数加入数据更新，即每次点击选择路径都能选择新文件获取新的执行数据。用户如果点击选择路径按钮并未选择文件时，通过 QMessageBox::warning（）提示相应警告信息如图 3-3-3。在数据分割时，把获取的数据同时显示 TableWidget 控件中，在 MainWindow 类的构造函数中设置 TableWidget 的列数，每列头行信息，同时添加滚轮控件以便数据过多需要进行翻页显示。根据获切割文本流读取到的信息，在通过选择文件按钮控件的信号对应的槽函数中给 TableWidget 设置确定行列的 TableWidgetItem 信息，具体实现效果实现如图 3-3-4。

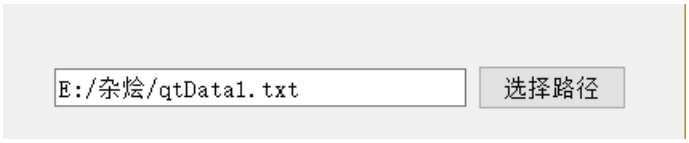


图 3-3-1（LineEdit 保存路径）

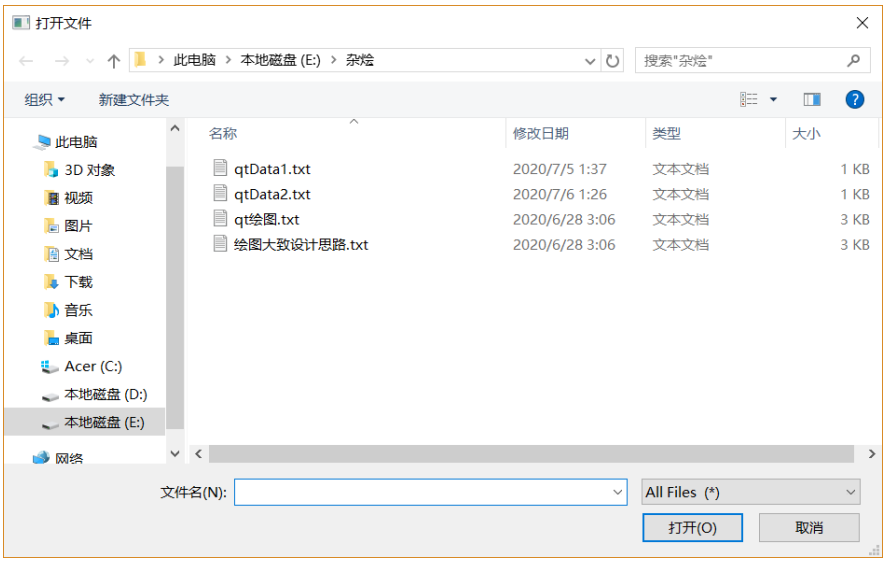


图 3-3-2（选择文件对话框）

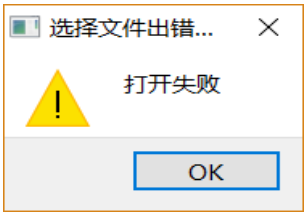


图 3-3-3(警告提示框)

	宽度	高度	重量	数量
1	41	12	1	1
2	25	34	1	1
3	19	44	1	1
4	115	22	1	1
5	25	51	1	1
6	16	22	1	1
7	71	22	1	1
8	44	109	1	1
9	41	29	1	1
10	90	87	1	1

图 3-3-4（TableWiget 显示读取到信息）

3.4 控件信号与槽函数的连接

通过 connect()函数基于函数指针的重载形式建立信号和槽的连接，由于在 Qt5.5 后支持 C++11 的 lambda 表达式，可以在关联时直接编写信号发射后要执行的代码，使用 Lambda 直接表达式写槽函数，让控件的信号与槽函数连接更加方便，不用在 MainWindow 类的声明中写上对每个函数具体命名，且加上关键字 SIGNALS。具体连接方式 connect(控件，控件的信号，执行的界面或控件，槽函数)。通过连接信号与槽函数的连接，可以槽函数修改静态的成员变量，从而达到用户可以通过界面交互形式的修改蚁群算法的信息启发因子，期望启发式因子，全局信息素挥发参数，局部信息素挥发参数,以及加工区域的宽、高和最大承重等信息。

3.5 绘图算法的实现

绘图需要调用 paintEvent 绘图事件来实现，需要 override 这个虚函数，最后调用系统提供 update()函数完成绘图事件的执行，达到绘图效果。首先需要将绘图坐标系的进行转换，没转换前绘制效果如图 3-5-1，需要将 X 轴方向反转，并将原点向下平移绘图画板长度的像素距离，就可以完成绘图坐标系的转换，由于绘制图形界面比较小，将绘制图形大小放大 2 倍，提高用户绘图体验。绘图需要的点坐标信息通过 Dialog 类下的 draw 函数来完成结果的更新。通过添加时间种子，以便获取随机颜色的 RGB 颜色的数值信息，同时获取到随机的颜色，用于填充绘制的方块区域。同时获取的将 RGB 颜色数值设置数据范围较小一些，防止 RGB 三色数值同时数值太大导致获取的颜色比较淡。具体加入绘图算法后绘制效果图 3-5-2。

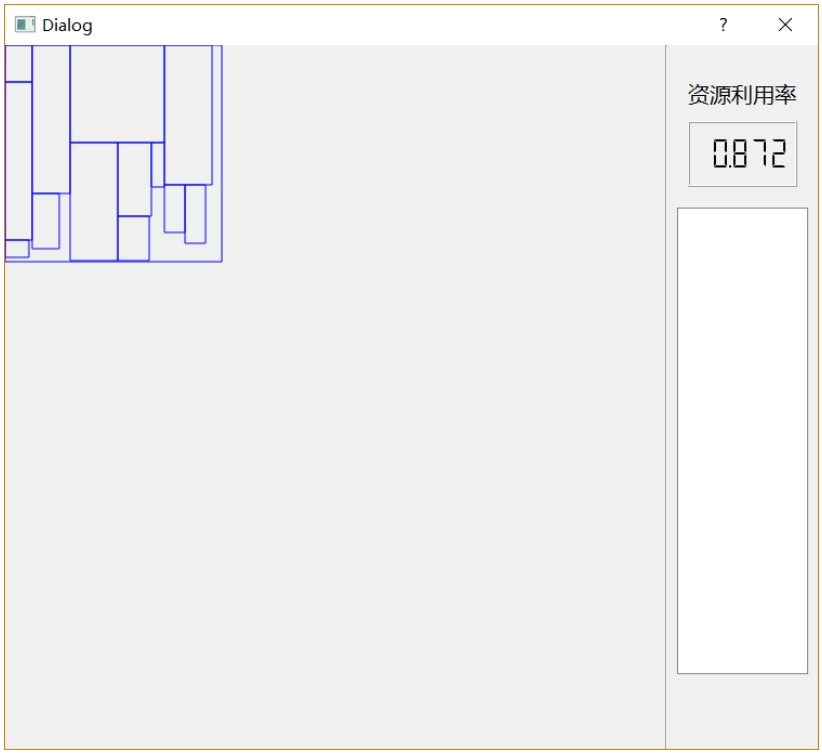


图 3-5-1（未加入绘图算法）

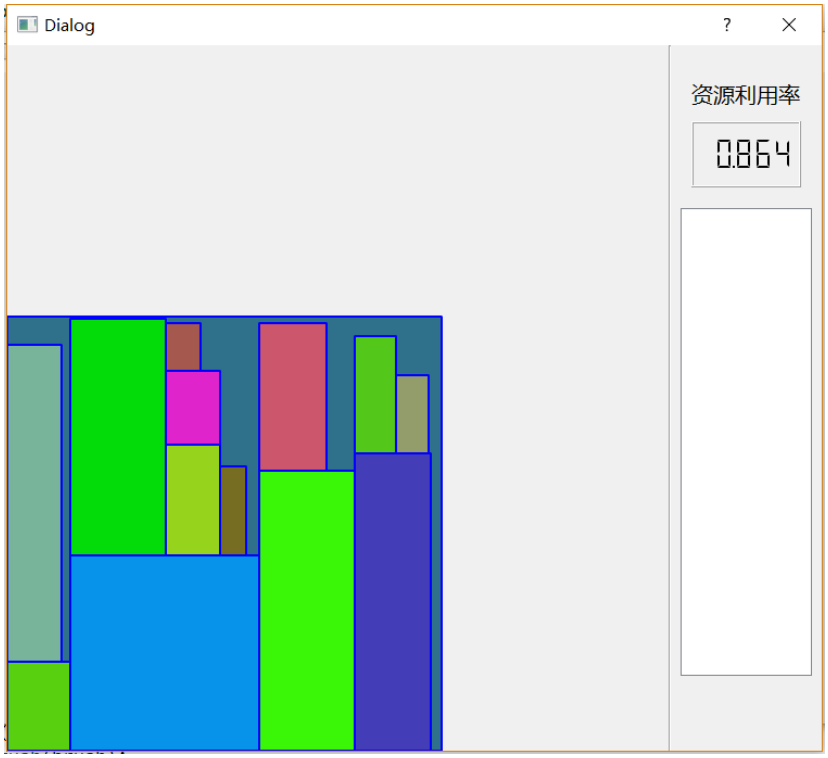


图 3-5-2（加入绘图算法后）

3.6 矩形约束限制算法

需要判断新出队列的材料是否能够放入空白区域，如果队列为空函数结束执行，否则分两种情况进行计算，第一种是新添加的材料就按照原本的形状不做出更改添加到加工区域中，如果不能放入将按照第二种情况尝试放入，第二种是将原来的加工材料进行转置，即把材料进行旋转 90 度横向进行添加到加工区域中。对这两种情况进行判断是否能将刚出队列的材料能否放入加工区域，将结果进行保存。此时已经将当前迭代情况具体哪些材料能放入到加工区域中都进行计算，并将结果进行保存。

4 主要函数说明

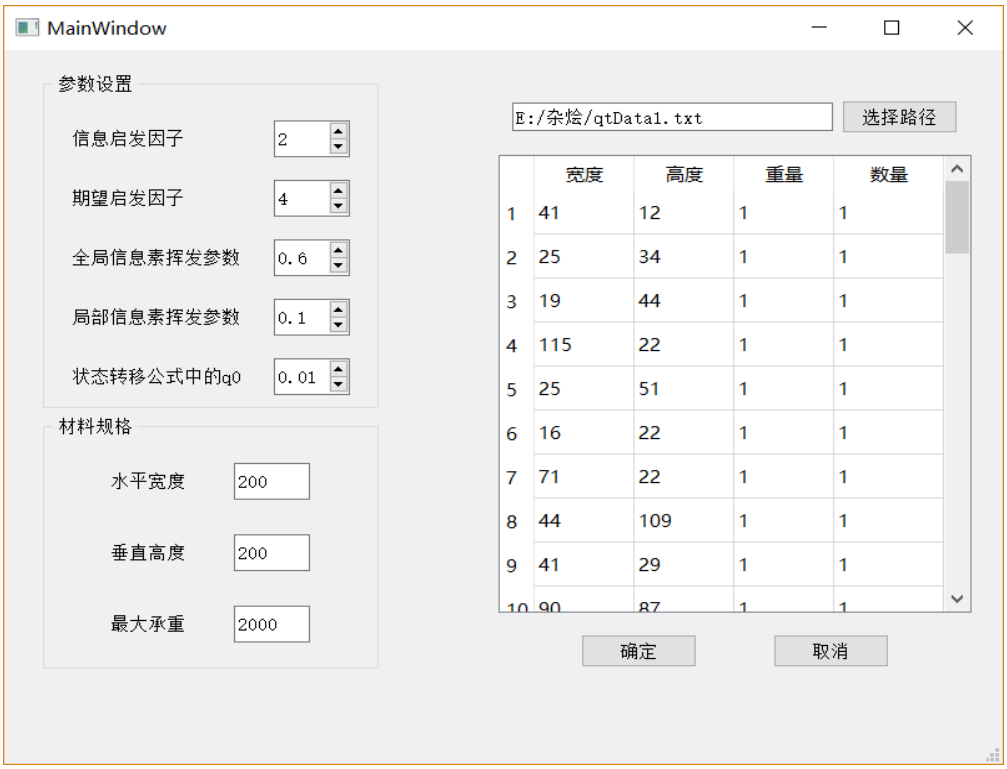
函数名称	函数功能
connect(ui->doubleSpinBox,&QDoubleSpinBox::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新信息启发因子
connect(ui->doubleSpinBox_2,&QDoubleSpinBox::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新期望启发因子
connect(ui->doubleSpinBox_3,&QDoubleSpinBox::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新全局信息挥发参数
c connect(ui->doubleSpinBox_4,&QDoubleSpinBox::textChanged,[=](){});;	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新局部信息挥发参数
connect(ui->doubleSpinBox_5,&QDoubleSpinBox::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新状态转移公式中的 q0
connect(ui->pushButton,&QPushButton::clicked,[=](){});	Lambda 表达式实现的槽函数用于用户需通过弹出选择文件对话框选择文件，将数据保存并显示
connect(ui->pushButton_2,&QPushButton::clicked,[=](){});	Lambda 表达式实现的槽函数用于用户点击按钮执行程序，并弹出绘图对话框
connect(ui->lineEdit_2,&QLineEdit::textCha	Lambda 表达式实现的槽函数用于用户界面

nged,[=](){});	交互式修改并更新加工区域的宽
connect(ui->lineEdit_3,&QLineEdit::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新加工区域的高
connect(ui->lineEdit_4,&QLineEdit::textChanged,[=](){});	Lambda 表达式实现的槽函数用于用户界面交互式修改并更新加工区域的最大承重
void MyDialog::display(double n);	显示资源利用率
void MyDialog::*paintEvent*(QPaintEvent *)	根据坐标点进行绘图
void MyDialog::draw( );	更新坐标点信息
int CLayout::CanBePutIn(CPart Part);	判断材料是否能放入加工区域，矩形约束

5 系统测试

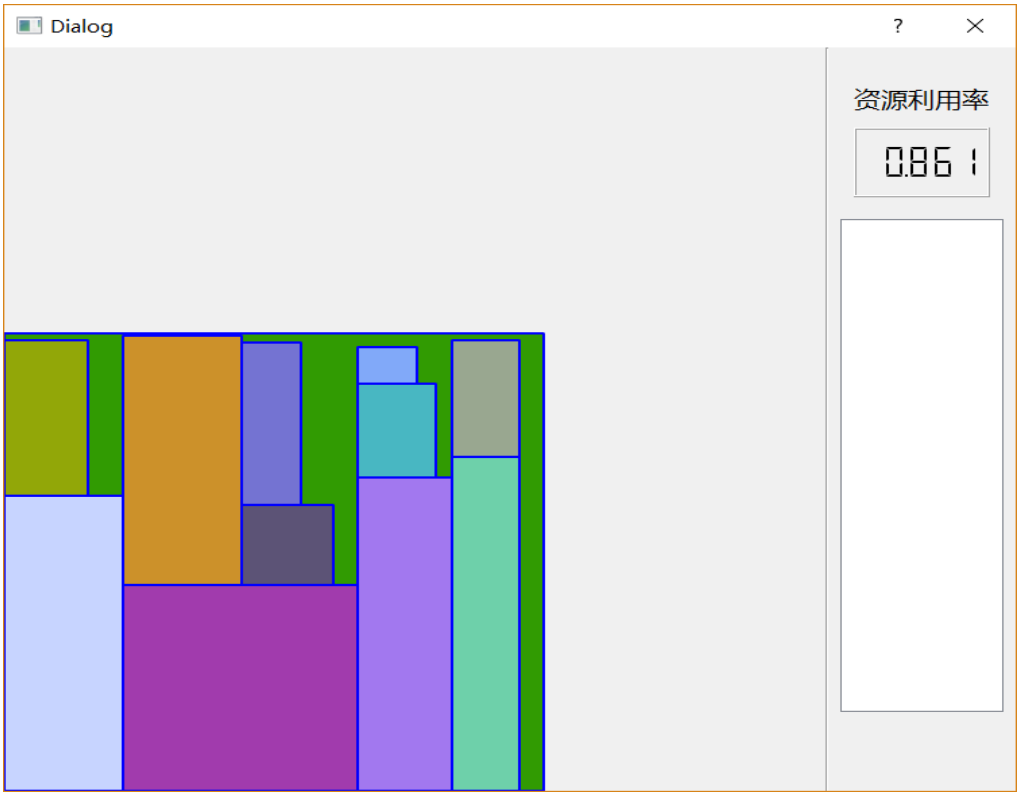
5.1 按文件路径存入零件数据

将零件信息通过文件的方式传入，包括零件的宽度，高度，重量和零件的数目，同时你也可以改变矩形加工区域的宽度，高度和承重上限。



5.2 对零件布局方案进行展示

对最优的摆放方案进行展示，将每一个零件的摆放以不同颜色的方式直观展示，同时显示当前方案的空间利用率。



附录：

6、源程序

1. MainWindow 类：

```
#include "CPart.h"
#include "CAreaList.h"
#include "CLayoutList.h"
#include "CLayout.h"
#include "ant.h"
#include <time.h>
#include <ctype.h>
#include <QMainWindow>
#include "mydialog.h"
QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:

    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    int ChooseNextNode(int currentNode, int visitedNode[],double
allDistance[partNum][partNum]);
    double CalAdjacentDistance(int node, caluDist &cd);

    static double alpha, beta, rou , alphal , qzero;
    //蚂蚁数量
    static int antNum;

    static double A[partNum][4]; //用于存放用户选择的文件的数据
private:
    Ui::MainWindow *ui;
    MyDialog *dialog;   //添加私有成员，为一个 dailog（窗口）的指针,用
于显示绘图界面
};
#endif // MAINWINDOW_H
```

MainWindow 类的源文件

```
#include <QFileDialog>  //文件提示框
#include <QMessageBox> //警告提示框
#include <QFile>  //创建保存文件地址的对象
#include <QFileInfo>  //文件信息文件
#include <QTextStream>   //文件按照文本流方式打开
#include <qDebug>      //跟 cout 类似，用于调试程序

#include "mainwindow.h"
#include "ui_mainwindow.h"
//信息启发因子，期望启发式因子，全局信息素挥发参数，局部信息素挥发参数，状态转移公式中的 q0
double MainWindow::alpha = 2 , MainWindow::beta = 4, MainWindow::rou = 0.6, MainWindow::alpha1 = 0.1 , MainWindow::qzero = 0.01 ;
double MainWindow::A[partNum][4] = {{0,0,0,0}};
int MainWindow::antNum = 15; //蚂蚁数量

int caluDist::AreaWidth = 200; //加工区域的宽
int caluDist::AreaHeight = 200; //加工区域的高
int caluDist::AreaWeight = 2000; //加工区域最大承重

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //确定水平方向上有多少列
    ui->tableWidget->setColumnCount(4);
    //设置水平方向的头
    ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"宽度"<<"高度"<<"重量"<<"数量");
    ui->tableWidget->horizontalHeader()->setStretchLastSection(true);

    for(int i = 0 ; i < 4; i ++ )ui->tableWidget->setColumnWidth(i,80);
    ui->tableWidget->setRowCount(50);


    //QspinBox  获取 alpha, 信息启发因子
    connect(ui->doubleSpinBox,&QDoubleSpinBox::textChanged,[=](){
        MainWindow::alpha = ui->doubleSpinBox->text().toDouble();
    });
    //QspinBox  获取 beta, 期望启发因子
    connect(ui->doubleSpinBox_2,&QDoubleSpinBox::textChanged,[=](){
        MainWindow::beta = ui->doubleSpinBox_2->text().toDouble();
    });
```

```
    //QspinBox  获取 rou, 全局信息素挥发参数,
    connect(ui->doubleSpinBox_3,&QDoubleSpinBox::textChanged,[=](){
        MainWindow::alpha = ui->doubleSpinBox_3->text().toDouble();
    });
    //QspinBox  获取 alpha1, 局部信息素挥发参数
    connect(ui->doubleSpinBox_4,&QDoubleSpinBox::textChanged,[=](){
        MainWindow::alpha1 = ui->doubleSpinBox_4->text().toDouble();
    });
    //QspinBox  获取 Qzero, , 状态转移公式中的 q0
    connect(ui->doubleSpinBox_5,&QDoubleSpinBox::textChanged,[=](){
        MainWindow::qzero = ui->doubleSpinBox_5->text().toDouble();
    });

    //蚂蚁数量
    /*connect(ui->lineEdit_2,&QLineEdit::textEdited,[=](){
       MainWindow::antNum = ui->lineEdit_2->text().toInt();
    }); */

    //加载文件
    connect(ui->pushButton,&QPushButton::clicked,[=](){

        QString path = QFileDialog::getOpenFileName(this,"打开文件
","E:\\杂烩\\"); //默认打开目录
        if(path.isEmpty())
        {
           QMessageBox::warning(this,"警告","打开失败");
        }
        else{
            //将路径 放入到 lineEdit
            ui->lineEdit->setText(path);

            //QFile File("E:\\杂烩\\qtData1.txt");
            QFile File(path);
            File.open(QFileDevice::ReadWrite);

            QTextStream in(&File);
            QString str; // 读取每行的三个信息
            partNum_t = str.count();  //获取, 每行个数
            QStringList strList =
in.readAll().split(QRegExp("[\r\n]"),QString::SkipEmptyParts); //截取
每行
            for (int i = 0; i < strList.count(); i ++ ) {
                str = strList[i];
```

```
                QStringList strlist_t = str.split(" ");
                ui->tableWidget->setItem(i,0,new
QTableWidgetItem(strlist_t[0]));
                ui->tableWidget->setItem(i,1,new
QTableWidgetItem(strlist_t[1]));
                ui->tableWidget->setItem(i,2,new
QTableWidgetItem(strlist_t[2]));
                ui->tableWidget->setItem(i,3,new
QTableWidgetItem(strlist_t[3]));

                A[i][0] =  QString(strlist_t[0]).toDouble();
                A[i][1] =  QString(strlist_t[1]).toDouble();
                A[i][2] =  QString(strlist_t[2]).toDouble();
                A[i][3] =  QString(strlist_t[3]).toDouble();
            }


        }


    });

    //更改矩形的大小  宽
    connect(ui->lineEdit_2,&QLineEdit::textChanged,[=](){
        caluDist::AreaWidth = ui->lineEdit_2->text().toInt();

    });
    //更改矩形的大小  高
    connect(ui->lineEdit_3,&QLineEdit::textChanged,[=](){
        caluDist::AreaHeight = ui->lineEdit_3->text().toInt();
    });
    //更改矩形的大小  承重
    connect(ui->lineEdit_4,&QLineEdit::textChanged,[=](){
        caluDist::AreaWeight = ui->lineEdit_4->text().toInt();
    });

    //执行确定按钮
    connect(ui->pushButton_2,&QPushButton::clicked,[=](){


        caluDist cd(A); //包含初始化测试数据


        int NcMax = 500;
```

```
int start = 1;
//绘图
dialog = new MyDialog(this);
dialog->setModal(true);
dialog->num = 0;  //注意要把，dialog 里面的数据清零


if (start)
{
    time_t timer, timerl;

    time(&timer);
    unsigned long seed = timer;
    seed %= 56000;
    srand((unsigned int)seed);

    //由矩阵表示两两城市之间的距离
    cd.calculateAllDistance();
    //蚁群系统对象
    AntColonySystem* acs = new AntColonySystem();
    ACSAnt* ants[antNum];
    //蚂蚁均匀分布在城市上
    for (int k = 0; k < antNum; k++)
    {
        ants[k] = new ACSAnt(acs, (int)(k % partNum));
    }
    cd.calculateAllDistance();
    cd.calculateAllWeight();
    //随机选择一个节点计算由最近邻方法得到的一个长度
    int node = rand() % partNum;

    AntColonySystem::totalCost = CalAdjacentDistance(node,cd);

    //各条路径上初始化的信息素强度
    acs->InitParameter(cd.testData,cd.allDistance);

    //全局最优路径
    int globalTour[partNum][2];
    //全局最优长度
    double gloalbestValue = 0.0;
    for (int i = 0; i < NcMax; i++)
    {
        //局部最优路径
        int localTour[partNum][2];
```

```
                //局部最优长度
                double localBestValue = 0.0;
                //当前路径长度
                double tourCost;
                double tourUtil = 0;
                for (int j = 0; j < antNum; j++) //每个蚂蚁行动
                {
                    CLayout Layout;
                    Layout.Init(caluDist::AreaWidth,
caluDist::AreaHeight, caluDist::AreaWeight, partNum);
                    CPart testCopy[partNum];
                    for (int i = 0; i < partNum; i++)
                    {
                        testCopy[i] = cd.testData[i];
                    }
                    int* tourPath = ants[j]->Search();
                    tourCost = cd.calculateSumOfDistance(tourPath,
tourUtil);                          //计算每个链的总价值和利用率
                    //tourCost = Layout.Calculate(tourPath, tourUtil,
testCopy);              //计算每个链的总价值和利用率
                    //tourWeight = calculateSumOfWeight(tourPath,
tourUtil2);//计算重量利用率
                    //qDebug() << "Main: tourWeight: " << tourWeight <<
endl;
                    //局部比较，并记录路径和长度
                    if (tourCost > localBestValue || abs(localBestValue
- 0.0) < 0.000001)
                    {
                        //qDebug() <<"Main: tourWeight: " << tourWeight
<< endl;
                        for (int m = 0; m < partNum; m++)
                        {
                            int row = *(tourPath + 2 * m);
                            int col = *(tourPath + 2 * m + 1);
                            localTour[m][0] = row;
                            localTour[m][1] = col;
                        }
                        localBestValue = tourCost;
                    }
                }
                //全局比较，并记录路径和长度
                if (localBestValue > gloalbestValue ||
abs(gloalbestValue - 0.0) < 0.000001)
                {
```

```
                    for (int m = 0; m < partNum; m++)
                    {
                        globalTour[m][0] = localTour[m][0];
                        globalTour[m][1] = localTour[m][1];
                    }
                    gloalbestValue = localBestValue;
                }
                acs->UpdateGlobalPathRule(*globalTour,
gloalbestValue);
                //输出所有蚂蚁循环一次后的迭代最优路径
                qDebug() << "Iterative optimal path " << i + 1 <<
localBestValue << "." << endl;
                for (int m = 0; m < partNum; m++)
                {
                    qDebug() << localTour[m][0] << ".";
                    //CargoNum[m] = 0;
                }
                qDebug() << endl;
            }
            //输出全局最优路径
            qDebug() << "Total value:" << gloalbestValue << endl;

            qDebug() << "Loading result:";
            double sum = 0;
            double sumWeight = 0;


            CLayout Layout;
            Layout.Init(caluDist::AreaWidth, caluDist::AreaHeight,
caluDist::AreaWeight, partNum);
            CPart copyData[partNum];//将原始数据复制到里面，不然放入的
时候会改变零件的数量
            for (int i = 0; i < partNum; i++)
            {
                copyData[i] = cd.testData[i];
            }

            int tour[partNum];
            for (int i = 0; i < partNum; i++)
            {
                tour[i] = globalTour[i][0];
                copyData[i].setId(i);
            }
            sum=Layout.Calculate(tour, copyData);
```

```
            qDebug() << endl;
            //AreaWidth*AreaHeight 背包容积
            qDebug() << "Total volume:" << sum << " Utilization ratio:"
<< sum / (caluDist::AreaWidth*caluDist::AreaHeight) << endl;
            qDebug() << "Total weight:" << sumWeight << endl;
            time(&timer1);
            double t = timer1 - timer;
            //qDebug() << testData[globalTour[0][0]].getHeight() << " "
<< testData[globalTour[0][0]].getWidth() << endl;

            //Layout.showLayoutList();

            // 绘图
            auto itLayoutList = Layout.m_lstLayout.begin();
            int grapWidth = 750, grapHeight = 650;
            dialog->resize(grapWidth,grapHeight); //设置绘图的基本宽
高


            dialog->init(0, 0, caluDist::AreaWidth,
caluDist::AreaHeight, grapWidth, grapHeight);

            //CRectangle panel(0, 0, panelWidth, panelHeight, grapWidth,
grapHeight); //创建平板车对象
            dialog->draw(); //画平板车 .

            for (; itLayoutList != Layout.m_lstLayout.end();
itLayoutList++)
            {
                dialog->init(itLayoutList->getX(),
itLayoutList->getY(), itLayoutList->getW(), itLayoutList->getH(),
grapWidth, grapHeight);
                //CRectangle part(itLayoutList->getX(),
itLayoutList->getY(), itLayoutList->getW(), itLayoutList->getH(),
grapWidth, grapHeight);
                dialog->draw();
                qDebug() << "X: " << itLayoutList->getX() << " Y: " <<
itLayoutList->getY() << " H: " << itLayoutList->getH() << " W: " <<
itLayoutList->getW() << " ";
                qDebug() << "Width: " <<
itLayoutList->getPart().getWidth() * itLayoutList->getHorizonAmount()
<< " Height: " << itLayoutList->getPart().getHeight() *
itLayoutList->getVerticalAmount() << " HorizonAmount: " <<
```

```
itLayoutList->getHorizonAmount() << " VerticalAmount: " <<
itLayoutList->getVerticalAmount() << endl;
                //swprintf_s(s, _T("%d"),
itLayoutList->getPart().getID());
                //int Id = itLayoutList->getPart().getID();
                //swprintf_s(s, _T("%d"), Id);

                //outtextxy(tranX + part.leftX + part.length / 3,
grapHeight - part.leftY - part.width/ 2 - tranY, s); //标注数字
                //i++;
                dialog->display(sum /
(caluDist::AreaWidth*caluDist::AreaHeight));
                dialog->update();

            }



            qDebug() << Layout.AvailSize() << endl;
            qDebug() <<"Number of parts:" <<Layout.getLayoutListSize()
<< endl;
            qDebug() << "time:" << t << endl;
            Layout.showLayoutPartNo();
            //dialog->settableWidget(Layout);
            dialog->show();
            //initgraph(640, 480);
            //setorigin(0,0);
        }



    });


}

MainWindow::~MainWindow()
{
    delete ui;
}
int MainWindow::ChooseNextNode(int currentNode, int visitedNode[],
double allDistance[partNum][partNum])
{
    int nextNode = -1;
```

```
    double shortDistance = 0.0;
    for (int i = 0; i < partNum; i++)
    {
        //去掉已走过的节点,从剩下节点中选择距离最近的节点
        if (1 == visitedNode[i])
        {
            if (shortDistance == 0.0)
            {
                shortDistance = allDistance[currentNode][i];
                nextNode = i;
            }
            if (shortDistance < allDistance[currentNode][i]) //
            {
                nextNode = i;
            }
        }
    }
    return nextNode;
}
//
double MainWindow::CalAdjacentDistance(int node, caluDist &cd)
{
    double sumCost = 0;
    int visitedNode[partNum];
    for (int j = 0; j < partNum; j++)
    {
        visitedNode[j] = 1;
    }
    visitedNode[node] = 0;
    int currentNode = node;
    int nextNode;
    do
    {
        nextNode = ChooseNextNode(currentNode, visitedNode,
cd.allDistance);
        if (nextNode >= 0)
        {
            //sum += cd.allDistance[currentNode][nextNode] -
cd.allDistance[currentNode][currentNode];
            sumCost += cd.testData[currentNode].getArea(); //fix: 改成
截止日期
            currentNode = nextNode;
            visitedNode[currentNode] = 0;
        }
```

```
    } while (nextNode >= 0);
    //sum += cd.allDistance[currentNode][node] -
cd.allDistance[currentNode][currentNode];

    //sumCost += C[currentNode][1];
    sumCost += cd.testData[currentNode].getArea();
    return sumCost;

}
```

2 .Dialog 类：

```
#ifndef MYDIALOG_H
#define MYDIALOG_H

#include <QDialog>
#include <QPainter>
#include <QPen>
#include <QColor>
#include "clayout.h"
const int Num = 50;

namespace Ui {
class MyDialog;
}

class MyDialog : public QDialog
{
    Q_OBJECT

public:

    int leftX, leftY;  //矩形左下角坐标
    int length, width; //矩形右上角坐标
    int orientedF; //是否是横向摆放，（对于初始情况来说）

    void init(int x, int y, int len, int wid, int w_hei, int w_wid);
    explicit MyDialog(QWidget *parent = nullptr);

    void draw();
    void display(double n);  //显示资源利用率
    void  paintEvent( QPaintEvent * );//自动调用 ，绘图
    static int num;
    int map[Num][5];
```

```
    ~MyDialog();

private:
    Ui::MyDialog *ui;


};

#endif // MYDIALOG_H
```

Dialog 类的源文件

```
#include "mydialog.h"
#include "ui_mydialog.h"
#include "QTimer"
#include <QTime>
#include <qDebug>
int MyDialog::num = 0;
void MyDialog::init(int x, int y, int len, int wid, int w_hei, int w_wid)
{

    leftX = x;
    leftY = y;
    length = len;
    width = wid;
    //orientedF = 0;
    w_width = w_wid;
    w_height = w_hei;
}

MyDialog::MyDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::MyDialog)
{
    ui->setupUi(this);
    qsrand(QTime(0,0,0).secsTo(QTime::currentTime()));//种下产生随机
数的种子
}

void MyDialog::draw( )
{
    map[num][0]= leftX;
    map[num][1]=leftY;
    map[num][2]=length;
```

```
    map[num][3]=width;
    num ++;

}

void MyDialog::display(double n)
{
    ui->lcdNumber->display(n);

}

void MyDialog::paintEvent(QPaintEvent *)
{
    QPainter painter(this);

    //坐标变换
    QTransform transfrom;
    transfrom.translate(0,650);
    transfrom.rotate(180,Qt::XAxis);
    transfrom.scale(2,2);

    painter.setRenderHint(QPainter::Antialiasing);
    QPen pen;

    pen.setBrush(Qt::blue);
    painter.setPen(pen);

    QTimer t;

    for(int i = 0 ; i< num ; i++ ){
        painter.setWorldTransform(transfrom);
        //产生随机颜色，不要产生颜色交淡的颜色
        QColor c(qrand()%200,qrand()%200,qrand()%200);
        QBrush brush(c);
        painter.setBrush(brush);
        painter.drawRect(map[i][0], map[i][1], map[i][2] ,map[i][3]);


    }

}


MyDialog::~MyDialog()
```

```
{

    delete ui;

}
```

3.CLayout 类

```
#pragma once
#define ANT_MAX_COUNT 50
#include "CAreaList.h"
#include "CLayoutList.h"
#include "CPart.h"
#include <list>
#include "mydialog.h"
#include <tchar.h>
using namespace std;
class CLayout
{

public:
    double m_dHeight;
    double m_dWidth;
    double m_dWeight;//重量
    //double m_dWeightLeft[ANT_MAX_COUNT];//重量剩余
    double m_dWeightLeft;//重量剩余
    double m_dUsedArea;
    int nPartsAmount;
    list<CAreaList> m_lstAvailable;
    list<CAreaList> m_lstUseless;
    list<CLayoutList> m_lstLayout;

    CLayout() {}
    ~CLayout() {}
    void Init(double dAreaWidth, double dAreaHeight, double
dAreaWeight,int nPartsAmount);//初始化区域
    int CanBePutIn(CPart Part);
    int PutIn(CPart& Part, bool bTrans);//返回装入的数量 引用为了修改
零件的剩余重量
    double UpdateList(double dWidth, double dHeight, int nHorizonAmount,
int nVerticalAmount, CPart Part, double dWeight);
    void Merge();
    void FindUseless(CPart* data);
    double Calculate(int* tour, CPart* Part);
```

```
    list<CAreaList>::iterator getAvaliableBegin() { return
m_lstAvailable.begin(); }
    list<CAreaList>::iterator getAvaliableEnd() { return
m_lstAvailable.end(); }
    void testMerge();
    void showLayoutList();
    void showLayoutPartNo();
    int getLayoutListSize() { return this->m_lstLayout.size(); }
    int AvailSize() { return m_lstAvailable.size(); }
};
```

CLayout 类的源文件（仅包含约束限制函数）

```
int CLayout::CanBePutIn(CPart Part)//这里没有考虑可用空间为 0 的情况
{
    //-1 转置 1 不转置 0 不能放入
    if (m_dWeightLeft < Part.getWeight() || m_lstAvailable.empty())
    {
        //cout << "CanBePutIn: WeightLeft: " << m_dWeightLeft << "
PartWeight: " << Part.getWeight() << " AvailableSize: " <<
m_lstAvailable.size() << endl;
        //if (m_dWeightLeft < Part.getWeight())cout << "重量不够不能放
置" << endl;
        //if (m_lstAvailable.empty()) cout << "没有可用空间" << endl;
        return 0;//如果重量放不下就返回 0
    }
    //cout<< " AvailableSize: " << m_lstAvailable.size() << endl;
    list<CAreaList>::iterator itAvail = this->m_lstAvailable.begin();
    double dPartWidth = Part.getWidth();
    double dPartHeight = Part.getHeight();
    int nVerticalMost, nHorizonMost;//竖直、水平方向最多放置
    int nWeightMost;
    int CanbePutInMost;
    int nHorizonReal;//水平实际放置几块
    int nPutInAmount = 0;
    double dSurplusAreaWithoutTrans = -1;//
    double dSurplusAreaWithTrans = -1;//转置放置上方剩余面积

    //不转置放入
    //cout << "CanBePutIn: WithoutTrans: AvailWidth: " <<
itAvail->getWidth() << " PartWidth: " << dPartWidth << " ";
    nVerticalMost = int(itAvail->getHeight() / Part.getHeight());
    //cout << "CanBePutIn: AvailHeight: " << itAvail->getHeight() << "
PartHeight: " << Part.getHeight() << endl;
    nWeightMost = int(m_dWeightLeft / Part.getWeight());
```

```
    CanbePutInMost = min(nWeightMost, Part.getSurplusAmount());
    //cout << "CanBePutIn: VerticalMost: " << nVerticalMost << "
WeightMost: " << nWeightMost << " CanbePutInMost: " << CanbePutInMost <<
endl;
    if ((nVerticalMost == 0) && int(itAvail->getHeight() /
Part.getWidth() == 0)) return 0;
    if (nVerticalMost > 0 && itAvail->getWidth() >= dPartWidth)
    {
        //cout << "不转置可以放入" << endl;
        if (CanbePutInMost <= nVerticalMost)
        {
            nPutInAmount = CanbePutInMost;
            dSurplusAreaWithoutTrans = dPartWidth *
(itAvail->getHeight() - nPutInAmount * dPartHeight);
            //cout << "可以直接放入一列 PutInAmount: " << nPutInAmount
<< " SurPlusArea: " << dSurplusAreaWithoutTrans << "垂直高度: " <<
nPutInAmount * dPartHeight << endl;

        }
        else
        {
            nHorizonMost = int(itAvail->getWidth() / dPartWidth);
            for (nHorizonReal = 1; nHorizonReal <= nHorizonMost;
nHorizonReal++)
            {
                if ((nHorizonReal * nVerticalMost) > CanbePutInMost)
break;
            }
            nHorizonReal--;
            dSurplusAreaWithoutTrans = (dPartWidth * nHorizonReal) *
(itAvail->getHeight() - dPartHeight * nVerticalMost);
            nPutInAmount = nHorizonReal * nVerticalMost;
            //cout <<"不能直接放一列 水平数量 " <<nHorizonReal <<" 竖
直数量 " <<nVerticalMost <<" SurPlusArea: " <<dSurplusAreaWithoutTrans
<< " 水平高度 " << nHorizonReal * dPartWidth << " 竖直高度 " <<
nVerticalMost * dPartHeight << endl;
        }
    }
    //else cout << "不转置不可以放入" << endl;
    int nPutInAmountWithoutTrans = nPutInAmount;
    //cout << "CanBePutIn: WithoutTrans: PutInAmount: " << nPutInAmount
<< endl;
    //转置后放入
```

```
    //cout << "CanbePutIn Before Swap PartHeight: " << dPartHeight << "
PartWidth: " << dPartWidth << endl;
    swap(dPartHeight, dPartWidth);
    /*dPartHeight = Part.getWidth();
    dPartWidth = Part.getHeight();*/
    //cout << "CanbePutIn After Swap PartHeight: " << dPartHeight << "
PartWidth: " << dPartWidth << endl;
    nVerticalMost = int(itAvail->getHeight() / dPartHeight);
    nWeightMost = int(m_dWeightLeft / Part.getWeight());
    CanbePutInMost = min(nWeightMost, Part.getSurplusAmount());
    /*cout << "CanBePutIn: WithTrans: AvailWidth: " <<
itAvail->getWidth() << " PartWidth: " << dPartWidth << " ";
    cout << "CanBePutIn: AvailHeight: " << itAvail->getHeight() << "
PartHeight: " << dPartHeight << endl;
    cout <<"CanBePutIn: VerticalMost: "<<nVerticalMost <<" WeightMost:
" << nWeightMost << " CanbePutInMost: " << CanbePutInMost << endl;*/
    if (nVerticalMost > 0 && itAvail->getWidth() >= dPartWidth)
    {
        //cout << "转置可以放入" << endl;
        if (CanbePutInMost <= nVerticalMost)
        {
            nPutInAmount = CanbePutInMost;
            dSurplusAreaWithTrans = dPartWidth * (itAvail->getHeight()
- nPutInAmount * dPartHeight);
            //cout << "可以直接放入一列 PutInAmount: " << nPutInAmount
<< " SurPlusArea: " << dSurplusAreaWithTrans << "垂直高度: " <<
nPutInAmount * dPartHeight << endl;
        }
        else
        {
            nHorizonMost = int(itAvail->getWidth() / dPartWidth);
            for (nHorizonReal = 1; nHorizonReal <= nHorizonMost;
nHorizonReal++)
            {
                if ((nHorizonReal * nVerticalMost) > CanbePutInMost)
break;
            }
            nHorizonReal--;
            dSurplusAreaWithTrans = (dPartWidth * nHorizonReal) *
(itAvail->getHeight() - dPartHeight * nVerticalMost);
            nPutInAmount = nHorizonReal * nVerticalMost;
            //cout <<"不能直接放一列 水平数量 "<<nHorizonReal <<" 竖
直数量 " << nVerticalMost << " 放入数量 " << nPutInAmount << "
```

```
SurPlusArea: " << dSurplusAreaWithTrans << " 水平高度 " << nHorizonReal
* dPartWidth << " 竖直高度 " << nVerticalMost * dPartHeight << endl;
        }
    }
    //else cout << "转置不可以放入" << endl;
    //cout << "CanBePutIn: WithTrans: PutInAmount: " << nPutInAmount <<
endl;
    //比较是否转置
    if (nPutInAmountWithoutTrans == 0 && nPutInAmount == 0) return 0;
    if (dSurplusAreaWithoutTrans <= 0 && dSurplusAreaWithTrans <=
0)return 0;
    else if (dSurplusAreaWithoutTrans <= 0 && dSurplusAreaWithTrans >
0)return -1;
    else if (dSurplusAreaWithoutTrans > 0 && dSurplusAreaWithTrans <=
0)return 1;
    //如果只放置一个 则随机决定是否转置
    else
        return dSurplusAreaWithoutTrans <= dSurplusAreaWithTrans ? 1 :
-1;
```

总结

在进行了两个星期的课程设计后，我对本学期所学习的《数据结构》和《算法设计与分析》这两门课的内容有了更深刻的理解和体会，对书上的理论知识通过这次课设有个很好的实战机会。这次和队友自拟题目，由于寒假在家自学了Qt的图形界面，我的主要任务就被分配为图形界面设计，刚开始的时候程序还没怎么开始做的时候，没法做界面时就帮助队友写一些约束判定函数，并一起讨论设计出便于绘图、便于界面传参和改参的数据结构的定义。如果用Qt绘图的话，必须用Qt提供的C++编译器去编译,但Qt的C++编译器与常规的C++编译器有些许不同,例如用qDebug()尽量代替cout,如果使用STL需要前面需要加上Q,例如vector需要变成Qvector,具体加不加可以查阅Qt的帮助文档，由于提前给队友做出提示和交流，让队友少走一些弯路快速适应Qt。这次最令我影响最深的就是要和队友多多交流,有问题一定要讨论遇见的问题,我们需要做出那些调整才能解决问题。由于上次C++课设我们还是我们仨人一组，对比上次课设，这次课设比上次课设效率高多了。主要上次课设我们虽说都在学校，但交流的太少了。这次课设虽说大家都在家，但是我们几乎每天都在线上开腾讯会议，讨论遇见的问题，报告自己任务完成的情况，并对任务分配进行合理的调整，以至于不会出现去一个人去等另一个人把代码写完，这样大大提高了我们这次课设的开发效率。我在课设中最大的收获，无论是算法还是数据结构，都需要通过实践才能掌握它最基本也是最重要的思想。同时我队友在课程设计的过程队友也都相当给力，由于考试时间压的太紧，做课设的时间不充裕，和队友都是一起熬夜赶代码，也都没什么怨言。即便熬夜赶代码，但是我们还是有一些原本计划好的没时间做，我的界面本来计划在绘图界面的对话框加上摆放在加工区域的材料的具体信息。可是后来实在没时间做了，就搁置了。

通过这次课设，我不仅学习到了理论知识，更加强了我的思维逻辑能力，这些能力的培养往往比今天学到的理论知识更对人有塑造作用。通过知识的累积我学会了如何更好地发现问题和解决问题,虽然在解决问题的过程中遇到了一些困难和阻碍,但是幸运的是我遇到了一群良师益友帮助我解决了问题让我对算法有了更深入的思考和了解。在以后的学习生活中，我会灵活运用在学习算法的过程中积累的经 验和方法，去解决遇到的阻碍或寻找一个问题的更优算法。

参考文献

[1]郑荣杰，张鹏程，崔海良,等. 基于蒙特卡罗方法的矩形布局问题研究[J]. 图学学报，2012，33(4):P.33-36.

[2]李士勇，陈永强，李研. 蚁群算法及其应用[M]. 哈尔滨:哈尔滨工业大学出版社，2004.

[3]汤安平，杨恢先，谭正华. 一种基于遗传算法的优化排料方法[J]. 计算机应用与软件，2009(01):171-172.

[4]陈文周.Qt5开发及实例[M].北京:电子工业出版社，2017

[5]安晓辉.Qt Quick核心编程[M].北京:电子工业出版社，2015

致　谢

这次的课程设计顺利完成，首先是邵虹老师的悉心指导的对我们问题的耐心的解答，让我们对算法和数据结构的选择上避免了很多错误，才能让我们在解决问题的时候用最低的空间和时间去得到最优的答案。同时感谢杨威和王宏生老师这一学期的栽培。最后当然也要感谢我的队友们，他们在课程设计中做出的贡献是无可替代的，有时他们不仅要解决自己遇到的问题，还要帮我解决遇到的问题，即使这不在他们的分工范围里。在期末这个阶段大家都很忙，但他们还是伸出了援手，这让我很感谢他们，没有他们，很多问题可能会解决，但不会这么快，大家有时候为了一个bug会改到天亮这直接去吃早餐，但没有人会抱怨，也没有人会去埋怨某个人，大家只想着去把这个项目做好，感谢他们的包容和对课程设计的热情，没有他们，课程设计的速度不可能这么快和这么好。

附录：

《数据结构与算法》课程设计成绩评定表

评价内容	具 体 要 求	分值	得分
平时表现	实验过程中，无缺勤现象，态度积极，具有严谨的学习态度和认真、踏实、一丝不苟的科学作风。	10	
实验内容	能够正确理解和掌握所涉及的概念和原理，合理的设计数据结构和程序结构，能够设计测试用例，运行结果正确，认真记录实验数据，原理及实验结果分析准确，实验具有一定的创新性。	60	
报告质量	实验报告格式规范，题目符合要求；报告内容充实、正确，实验总结归纳合理到位。	30	
总 分			