贾贷锡 于兴国 赵泉斌

计算导论程序概要设计

简介

此次概要设计为Version 2.1 版本，包含产品处理和动画输出两方面内容，组别为306班4组，成员为贾岱锡、于兴国、赵泉斌三人。

## 产品处理

与Version 1.1 版本相同，均采用了模拟退火算法，本次对模拟退火算法进行了一系列的改进，以便获取更优解。

## 动画输出

动画输出部分采用了教师课上给出的EGE方案，动画输出部分由于是此次新添加的，所以此版本对EGE图形的设计占有主要部分，此版本概要设计主要对此部分进行介绍。

1. **产品处理的改进**

模拟退火算法在调试参数时比较复杂，由于本组的Version 1.0 版本并未使用模拟退火，而是单纯利用调度规则来进行求解，显然，得到的结果并不理想，非常容易陷入局部最优解，且给定的结果一旦确定，得到的结果就一定是确定的。

模拟退火算法摒弃了调度规则的确定性，尝试使用随机得到一组解的方式跳出局部最优解，以便获得全局更优解。在 Version 1.1 版本的改制过程中，由于剩余时间较短，算法处理比较粗糙，对于这类问题的解优化并不明显。通过近阶段的集体讨论，对模拟退火算法进行了以下优化：

1. 优化了获取新解的方式

模拟退火算法的精髓之处在于以一定的概率接受恶化解，但由于本程序涉及到300s 的时间限制，有一定的概率等程序运行完毕（300s 时间到）后，得到的最终解会差于在程序运行过程中随机跑出的解，所以我们在程序中添加了“最小值计数器”，即用最终得到的解与程序运行过程中的最小值进行比较，如果最终得到的解差于最小值计数器中的值，则替换程序运行的最终解，这样可以优化结果。

for(i = 0; i < L; i++)

{

re\_new = create\_new(n, re);

time\_new = get\_time(j, m, n, re\_new);

time = get\_time(j, m, n, re);

if(time\_new < time\_min){

time\_min = time\_new;

for(k = 0;k < n;k ++)

re\_min[k] = re\_new[k];

}

1. 优化了迭代参数

模拟退火算法要进行大量的循环操作，随着循环次数的增多，结果逐渐趋于最优解，所以循环参数的设置就显得尤为重要，经过大量的数据测试，并考虑到充分利用300s 的时间，此版本设置了如下参数：

T0 : 10000000.0 //初始温度

T\_END : (1e-8) //结束温度

Q : 0.98 //退火系数

LIMIT : 300000 //最长运行时间

df //产生新解与当前解的差

exp(-(double)df/T) //接受恶化解的概率

3. 优化了与动画输出的衔接

为了动画输出能够获得更好的交互体验，我们进行了产品处理与动画输出的衔接操作，使得我们最终获取的结果能更好的被动画输出部分调用。我们既继承了代码的“高内聚，低耦合”的特性，又使得程序的两大部分易于交互链接。我们充分利用了output.txt中的数据，并只使用output.txt中的数据就完成了动画部分的输出工作，真正实现了“高内聚，低耦合”。

**二、动画输出的处理**

动画输出我们使用了老师提供的EGE方案，虽然发现了其中的一些问题，但最后还是较好的完成了既定任务。下面简述动画实现的操作过程：

1. 数据获取

我们通过从output.txt中获取所需数据，将数据存储到了一个名为coordinate\_0[][]的二维结构体数组中，里面存储了机器、工件、时间等数据，为了甘特图的绘制、检修等操作提供了充足的数据支持。

数据的获取涉及到文件的操作，具体实现代码如下：

void get\_struct**()**

**{**

int i**,**j**;**

int flag **=** 1**;**

char ch**;**

FILE**\*** fptr **=** fopen**(**"output.txt"**,**"r"**);**

**if(**fptr **==** **NULL){**

printf**(**"The file could not be opened!\n"**);**

exit**(**EXIT\_FAILURE**);**

**}**

**for(**i **=** 0**;** flag **;**i **++)**

**{**

**for(**j **=** 0**;** ch **!=** '\n' **;**j **++)**

**{**

ch **=** fgetc**(**fptr**);**

**if(**ch **==** 'M'**)** fseek**(**fptr**,**2**,**SEEK\_CUR**);**

**else** **if(**ch **==** ' '**)** fseek**(**fptr**,**1**,**SEEK\_CUR**);**

fscanf**(**fptr**,**"%d"**,&**coordinate\_0**[**i**][**j**].**start**);**

fseek**(**fptr**,**1**,**SEEK\_CUR**);**

fscanf**(**fptr**,**"%d"**,&**coordinate\_0**[**i**][**j**].**work**);**

**if(**coordinate\_0**[**i**][**j**].**work **>** WORK\_NUM**)** WORK\_NUM **=** coordinate\_0**[**i**][**j**].**work**;**

fseek**(**fptr**,**1**,**SEEK\_CUR**);**

fscanf**(**fptr**,**"%d"**,&**coordinate\_0**[**i**][**j**].**step**);**

**if(**coordinate\_0**[**i**][**j**].**step **>** STEP\_NUM**)** STEP\_NUM **=** coordinate\_0**[**i**][**j**].**step**;**

fseek**(**fptr**,**1**,**SEEK\_CUR**);**

fscanf**(**fptr**,**"%d"**,&**coordinate\_0**[**i**][**j**].**End**);**

fseek**(**fptr**,**1**,**SEEK\_CUR**);**

ch **=** fgetc**(**fptr**);**

**}**

ch **=** fgetc**(**fptr**);**

**if(**ch **==** 'T'**)** **{**

flag **=** 0**;**

**while(**ch **!=** '\n'**)**

ch **=** fgetc**(**fptr**);**

fseek**(**fptr**,**9**,**SEEK\_CUR**);**

fscanf**(**fptr**,**"%d"**,&**END\_TIME**);**

**}**

**else** fseek**(**fptr**,-**1**,**SEEK\_CUR**);**

**}**

fclose**(**fptr**);**

**}**

2. 甘特图的绘制

甘特图的绘制相对简单，但需要从0 开始学习EGE中的相关操作，为了用于处理不同的工件，我们的程序提供了十六种（包括检修）配色，以便获得最好的视觉效果。

检修：#DC143C 猩红

工件配色方案：

#D8BFD8 0 蓟色

#BA55D3 1 适中的兰花紫

#8A2BE2 2 深紫罗兰色

#00008B 3 深蓝色

#B0C4DE 4 淡钢蓝

#5F9EA0 5 军校绿

#2F4F4F 6 深石板灰

#7FFFAA 7 绿玉色

#2E8B57 8 海洋绿

#FFFF00 9 纯黄色

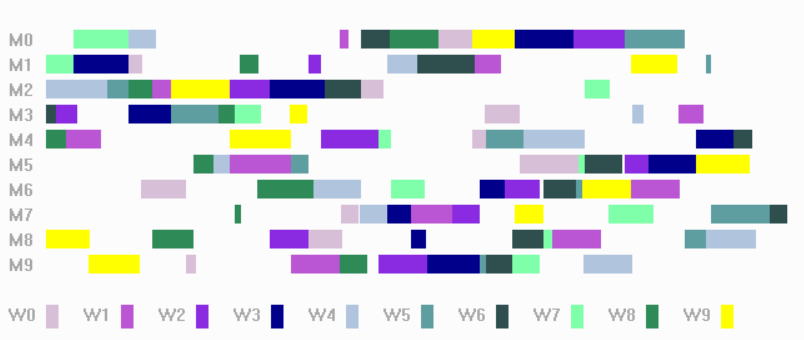
#F0E68C 10 卡其布色

#FFA500 11 橙色

#8B4513 12 马鞍棕色

#808080 13 灰色

#FFDEAD 14 纳瓦霍白

处理一组数据，得到的效果如下：

3. 处理检修

检修的调度规则本组采用了相对简便的平移操作，但平移过程中需要讨论多种情况：①检修点对应机器的工作状态②检修时长对下一步的影响③此次检修时长内又加入了其他检修情况。综上考虑我们设计了程序，得到了如下效果图：

4. 处理对话框

我们在设计对话框时没有用到线程操作，因为在EGE中加入线程操作处理起来会变得复杂，而用kibit()函数检测EGE是否有键盘键入，如果有键盘操作则弹出对话框，键入检修指令：