关于“task_organizer”的作业报告

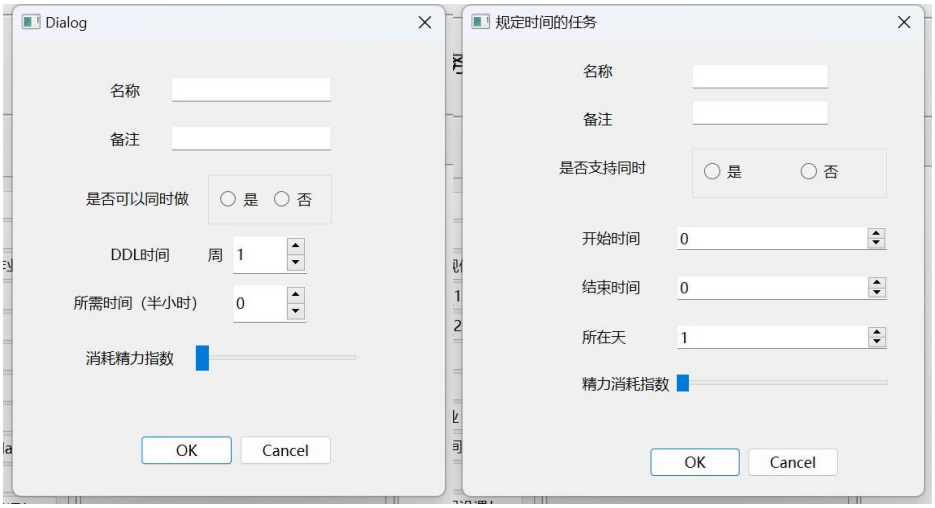
组名：我们仨——BWZ

一、主要功能

我们小组的 Qt 项目，是希望设计一个应用程序，为用户提供一周内个性化的任务安排建议，帮助用户更好地安排自己的事件，管理任务，完成 ddl。

（一）添加任务：

我们把常见的任务类型分为了 ddl 任务和定时任务，也就是有确定的 deadline 的任务，和有确定时间段的事件，用户可以选择添加这两种任务，并提供这些任务的具体信息：

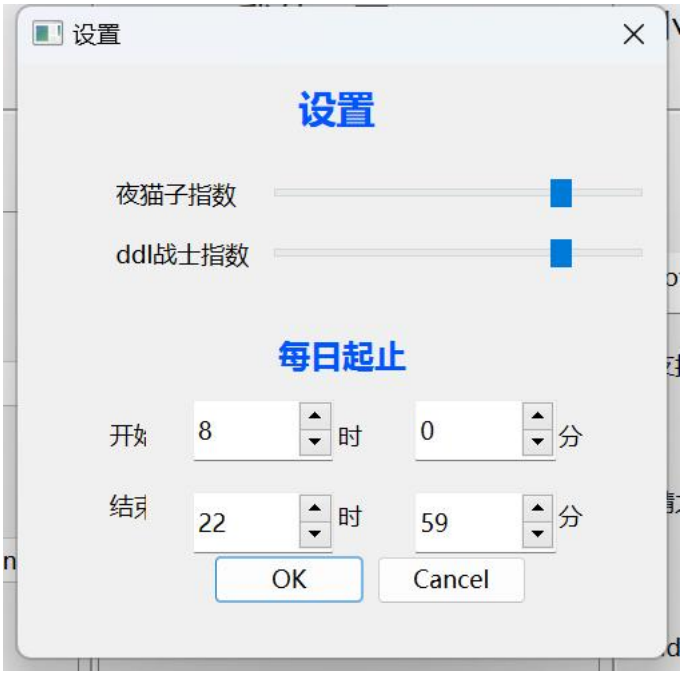


当然用户需要提供 ddl 任务预估的完成时间，定时任务的具体时间，以方便安排任务。另外我们希望用户能用最少的时间取得更好的效果，因此我们也会推荐用户在同一个时间段完成两件任务，这个需要用户来自己衡量某个任务能否和别的任务同时做，同时我们会优先推荐精力消耗少的任务安排到一起。新添加的

任务会被放到“待安排”栏里，并且任务的具体信息可以在右边查看或修改。

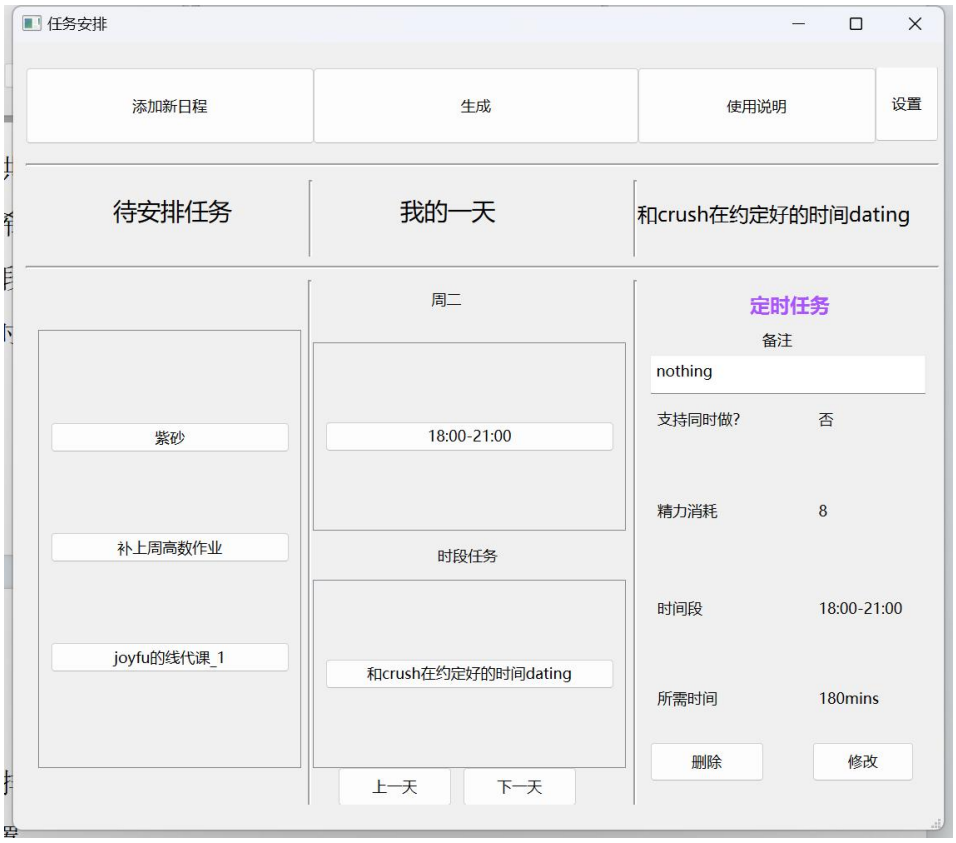
（二）个性化设置

由于不同的用户有着不同的习惯，有些人是 ddl 战士，习惯拖到最后，有些人是夜猫子，更喜欢在晚上干活。在“设置”里面，用户可以调整自己的这两种习惯，并且可以设置能够接受的最早时间和最晚时间。我们的任务安排也会根据这些个性设置做出调整



（三）生成任务安排

点击生成，即可获取任务安排。在 ddl 过多无法合理安排时间时，我们会给出警告弹窗，在“我的一天”栏里可以查看各个时间段的任务，以及其具体信息。如果修改任务信息，由于任务的 ddl 等信息被修改可能会影响任务安排，我们便将修改过的任务放回“待安排”栏里。同样可以删除某个特定任务。



二、设计细节

（一）信息型的类设计：

为了统一任务安排，我们设计了以下几个类：

Task：基础的类，存储任务的具体信息。通过 label、state 等区分任务的类型、安排状态；

AlltaskRecorder：统一记录所有任务的信息，每个任务有它独特的 id，并通过它来读取、储存、修改任务的具体信息，并且实现了从文件读取和输出到文件的接口。

Block：时间块，存储这个时间段内任务的 id（可能有一个或两个），这个时间段的具体起止时间、所在天；

Day：天，记录一周七天所有的 block；

TaskArranger：记录任务安排，其中存储的是一周内的 7 个 day，并提供了和文件的接口。

PersonalSet：记录个性化安排，包括 ddl 战士、夜猫子、起止时间，并提供和文件的接口。

（二）安排算法介绍：

我们的算法主要可以理解为两种需求，两个部分：

（1）**需求一**：启发于现实生活，我们希望能够实现任务的同时完成性。比如，当在完成用时 5 小时的跑代码任务时，我们可以在此期间同时进行其他的任务，如复习线性代数。

这一部分算法的**意义**在于，同时实现低难度的任务，我们的推荐安排就能实现在时间上的充分利用。

为了实现这一特性，我们设计了第一部分的算法：get_best_combination。这一算法的**功能**是：对用户放入的任务集进行处理，获得最好的任务组合：一系列 block。每个 block 表示是我们推荐的同时进行的任务。

推荐的原理：

（a）基本原理是：根据任务的难度和用时，获得最高效的任务组合。衡量的依据是：总 j_value（由组合难度和用时计算而出）越大，说明这个安排越高效。基于同时完成任务所需的精力消耗要大于单独分别完成两种任务的精力耗费这一事实，我们使用指数函数来进行价值估量，具体实现如下：

```
double j_value(const Task t1,const Task t2){
    return -pow(1.05,t1.hard+t2.hard)*max(t1.worktime,t2.worktime);
}
```

hard 属性：表示这一任务的难度，用来表示该任务的耗费精力程度

worktime 属性：表示任务的完成时间。

（b）特殊要求：

1、任务具有 sametimeable 属性：表示这个任务是否支持与其他任务同时做。对于值为 False 的任务，不参与任务安排。

2、当两个任务都完成时，这一任务组合才算完成。

3、如果定时任务 Fixed_task 参与组合，组合的时间必须包含该定时任务的时间区间。

算法实现：该推荐算法 get_best_combination 主要借助函数 combination_dfs 实现，该函数基于深度优先搜索，通过估值函数 j_value 从而获得最优的任务组合。具体的实现详见代码包，位于 task_arranger 文件中。

（2）**需求二**：我们希望能够根据用户的个性化偏好，将任务安排在最符合用户

偏好的时间段，从而实现最优的任务安排。

该需求由算法 time_arrange 进行实现。对第一个算法获得的一系列 block 进行操作。

算法实现：

（a）对于定时的 block（包含定时任务的 block 具有该属性），按其固定时间进行安排即可；

（b）（1）对于其他 block:依据 ddl 从周一到周日依此考虑。具体而言，对于 ddl 在周一的所有 block，可选空间为除去定时任务的周一全天，找出其最优安排；对于 ddl 在周二的所有任务，可选空间为除去定时任务、已安排的任务的周一、周二全天，找出其最优安排……依次进行，即可找出一周的最优安排。

（2）据可选空间找出最优安排由函数 dfs 实现。同样是基于深搜原理，估值函数为 get_value,刻画该安排与用户个性化需求的相符程度，实现如下：

```
double get_value(Block Block,double ddlwarrior, double nightworker) {

    return -abs(Block.endtime / 48.0 - nightworker)
           + -abs((double(Block.day * 48 + Block.endtime)) / double((Block.ddlday + 1) * 48) - ddlwarrior)*2;
}
```

总的来说，我们通过第一个算法实现任务的同时进行性，进而归化为 block，从而在第二个算法中实现对 block 的单独安排（block 的安排中一个时间段中只能有一个 block，这样就实现了向任务不能同时完成这一简单情况的转化），这样，就能找到最高效、最个性化、最贴切的最优任务安排。

（三）窗口：

基于功能分类，我们设计了以下几个窗口，并借助 Qt 的信号与槽机制实现上述功能：

add_new：选择添加任务的种类；

add_fixed_ddl：添加 ddl 任务具体信息；

add_fixed_time：添加定时任务具体信息；

settings：修改个性化设置；

explaining：提供使用说明；

mainwindow：主窗口，从 AlltaskRecorder、TaskArranger 等类中获取并显示

具体信息，获取各个子窗口的信号并进行相应操作。

三、任务分工：

主体框架的设计与实现：毕晨博；

安排算法的设计与实现：王辰硕；

用户界面的设计与美化：赵烁然。

四、项目总结与反思：

在这次的合作项目中，我们遇到了许多困难，比如算法与几个主体类的接口不一致，窗口关系的 bug 等，走了不少弯路，但是通过合作我们一一解决了诸多问题。事实上，在合作任务中，重要的不只是各自的能力，还有合理的任务安排、各自的沟通与交流。