循环赛工具

roundRobin_multitask_database.py

代码分析报告

目录

---

0. 多局比赛

0.1. 简介

比赛内核代码 match_core 实现了两方 AI 进行一次比赛的功能，并可保存双方函数存储空间为全局变量 STORAGE，在多局比赛之间保留；然而，由于多局比赛时需要涉及到初始化存储空间、交换场地等高于单局比赛尺度的操作，match_core.py 中未对其进行直接的实现。

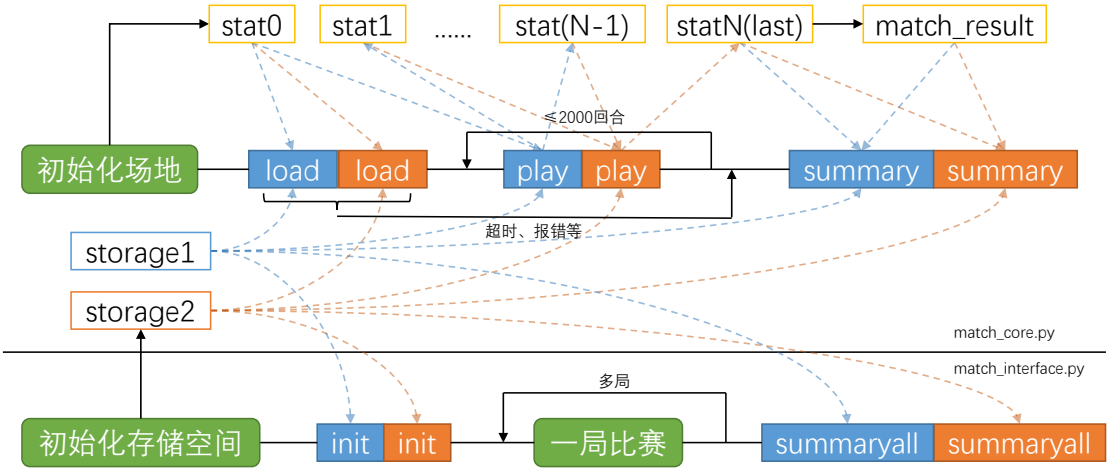

图 1 多局比赛流程图与参数传递

match_interface.py 中提供了多局比赛函数 repeated_match 对多局比赛的逻辑结构作示例，如上图所示包含了额外的操作与接口；本项目中未直接调用该函数，但使用了类似的逻辑结构运行多局比赛与传递参数。

0.2. 多轮比赛新增接口

在进行多轮比赛时，比赛框架为每个参与者提供了新的接口 init 与 summaryall，用于在进入比赛前接收存储空间，进行必要的数据导入与比赛总结等操作。

若 init 与 summaryall 函数未定义或报错将被跳过，不影响比赛进程。

接收参数：

函数存储空间 storage

1. 循环赛赛制介绍

分组小组赛阶段采用了循环赛赛制，小组每个参与者都与其余所有对手进行比赛，并决出胜负。

1.1. 比赛方式介绍

除一局比赛所需的参数 k、h、maxturn、maxtime 外，多局比赛额外定义了比赛回合数参数 rounds。

对小组赛中每个对手组合 plr1 与 plr2，将进行至少 rounds 场、最多 2*rounds 场比赛，其中前 round 场比赛由 plr1 先手、plr2 后手，后 rounds 场比赛由 plr2 先手、plr1 后手；若任意玩家总获胜次数大于 rounds，则提前结束比赛。

一个对手组合的胜负根据双方总获胜次数决定：若一方获胜场次大于另一方，则获胜方（次数多者）记 3 分，另一方记 0 分；若双方获胜场次相同则各记 1 分。

此外，为防止一方 AI 出现无限循环，每个对手组合的对决被设定了总时长阈值，其值为双方均达到最大思考时长且进行最大局数对决，并计入系统耗时的理论总时长。超过该时长的对手组合比赛将被强制结束，双方记 0 分。

1.2. 实现细节

生成赛制前，比赛框架首先将读入比赛目录内的所有玩家代码文件，验证其合法性并将文件名加入合法玩家列表。合法玩家列表将记录为 players.txt，导入错误信息将被记录为 errors.txt。

在获取玩家列表后，比赛框架将以该列表分别作为两方玩家 plr1 与 plr2，进行一次倒三角形的遍历，随机将(plr1, plr2)或(plr2, plr1)加入赛程列表；随后将使用 random.shuffle 打乱赛程列表，并对玩家列表按字典序排序，完成随机赛程的生成。赛制列表记录为 task_buffer 变量。

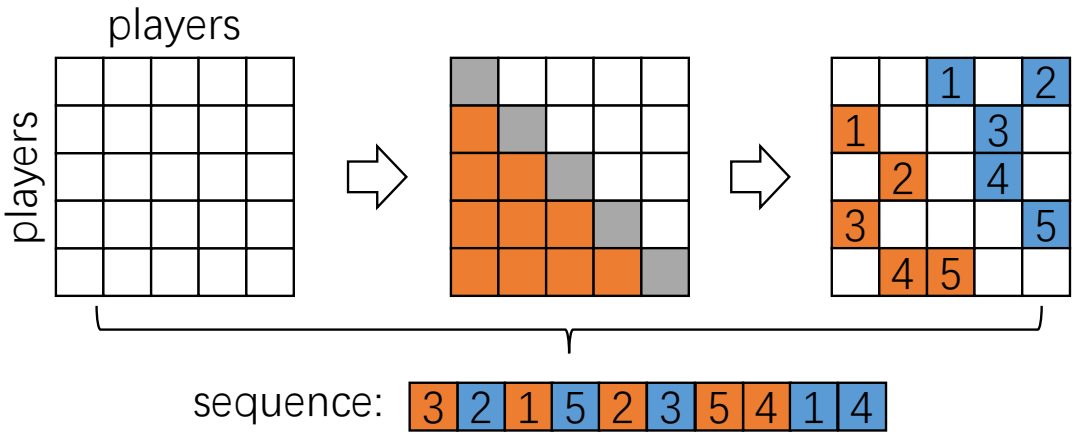

图 2 生成随机赛程序列

2. 代码结构

由于任务明确，本项目主进程中主体代码采取了顺序式而非模块化结构，拥有先后逻辑联系的代码块包含在 if __name__ == '__main__'语句之下；由于多进程启动时需要独立的内存空间，进程函数及其依赖的库（match_core，match_interface）位置在主进程代码块之前。

项目结构如下：

> 进程执行函数 process_task
> if __name__ == “__main__”:
>> 设置相关参数
>> 初始化数据库、进程池、进程数据传输队列、比赛结果统计等
>> 读取玩家代码、生成赛程序列
>> 定义数个辅助函数 **helpers**
>> **进入主事件循环**
>> 总结比赛、打包比赛记录

以下将介绍主进程的常量与数据结构，及着重介绍辅助函数与主事件循环的内容，其余部分可分别参考相应章节。

2.1. 设置参数

变量名	**内容**
MATCH_PARAMS	传入 match 函数进行单局比赛的设置参数
ROUNDS	比赛局数（单向），每对玩家比赛最大局数为二倍该值，任一玩家获胜局数超过该值判为胜利
MAX_TASKS	最大并行进程数（设置为 cpu 核数-2）
TIMEOUT	每个进程最大运行时间，由上述参数计算得到
MAX_PROMOTION	（显示）最大晋级人数，默认为 8，可读取命令行参数 2
TEAM	循环赛组别，读取命令行参数 1 得到；默认为 8
LOG_FORMAT	与 TEAM 变量相关的比赛记录保存路径，在进程内调用 比赛记录将保存为 log_format % (*names, index)路径
AI_PATH	由 TEAM 变量生成的代码文件夹绝对路径

2.2. 数据存储对象

变量名	**类型**	**功能**
dataq	多进程队列	由各比赛进程传出的单局比赛结果，由主进程接收
db	SQL 指针	将比赛记录写入对应组别的数据库
running_tasks	列表	记录当前运行的所有比赛选手名称、进程及其启动时间
rounds_stat	字典	通过嵌套字典记录每个玩家组合当前比赛结果 关键字：(先手玩家名,后手玩家名) 内容：{0:先手获胜次数,1:后手获胜次数,None:平局次数}
results_stat	字典	通过字符串记录已完成的对局结果 关键字：(先手玩家名,后手玩家名) 内容：”**+**”（先手胜）；”**-**“（先手负）；”**x**”（超时终止）

2.3. 辅助函数 helpers

在进入主循环前声明了数个辅助函数，用于简化代码内容、明确代码逻辑。

flush_queue()

清空进程队列 dataq 内待处理的单局比赛结果，将结果累加至 rounds_stat 字典，并计入数据库 match_raw 表

update_pair(names, flag)

在一对多轮对决结束（决出结果或超时终止）时运行，根据 rounds_stat 中相应关键字双向结果（name 与 name[::-1]）写入数据库并统计结果，将总结果写入 results_stat 相应位置，并将统计中间结果计入 match_result 表

参数：

names – 对决双方玩家名

flag – 结束类型（"FINISH"或"TIMEOUT"）

visualize(file=sys.__stdout__)

将当前比赛过程可视化至屏幕，绘制表格，统计各玩家总得分并排序显示于屏幕。根据设置的 MAX_PROMOTION 参数将在该排名玩家后方画线。

参数：

file – 输出流，默认为标准输出（屏幕），可输出至文件。

2.4. 主事件循环

主事件循环以时间驱动，使用了 time.perf_counter 作为获取当前时间的方式。其单次循环的内容按顺序如下：

a. 清空并统计当前的比赛结果队列
b. 遍历 running_tasks 列表，对其中每个任务，如果其执行完毕就标记 FINISH，否则如果其启动时间与当前时间之差高于阈值就标记 TIMEOUT；对有标记的任务对应的双方名称与标记传入 update_pair 函数汇总结果，并将其移除
c. 若赛程序列 task_buffer 非空，且当前运行任务数小于 MAX_TASKS，则取其中一个传入比赛进程运行，并将该赛程双方名称、该进程与当前时间加入 running_tasks 列表
d. 若距上次刷新可视化结果长于 0.5 秒，则刷新可视化界面并更新可视化计时
e. 在 running_tasks 列表为空时跳出循环

3. 多进程结构

3.1. 简介

注意到赛程中每个对手组合所进行的比赛是互相独立的，不受已完成的多轮比赛与结果的影响，在按照赛制运行比赛时采用多进程结构，可以充分利用比赛服务器多核 CPU 的计算力，大幅压缩比赛总时长。

3.2. 结构与算法设计

由于赛程内比赛组数往往远高于 CPU 核数，需要有效的方式维护运行的线程，使比赛进程中并行任务数维持在合理的范围内；此外，由于存在强制结束超时进程的需求，multiprocessing.Pool 不支持该应用情景，本项目使用了 multiprocessing.Process 启动每个比赛单元，并在主进程内使用列表结构配合 time.perf_counter 进行时间控制。

与多进程的维护相关的代码包含在主进程的主事件循环中，其实现方式在“主事件循环”部分介绍。

4. 比赛记录

循环赛中每一局的比赛记录将输出为复盘记录 zlog 文件，同时比赛结果的简单统计也将保存在数据库中以备使用。

4.1. 原始比赛记录

match 函数返回的比赛原始记录将以 zlog 格式存储，其过程为使用 pickle 库将比赛记录对象序列化，之后使用 zlib 库对其进行压缩减少大小。一场典型的对局，其 zlog 记录通常为数 MB 大小。

zlog 文件可用可视化工具 visualize.py 打开并查看复盘记录。

4.2. 数据汇总与可视化

在比赛过程中，主进程执行了汇总统计比赛结果的任务，将由比赛进程经数据队列 dataq 传出的比赛结果计入 rounds_stat 字典，并在每组玩家比赛结束后汇总 rounds_stat 内结果并计入 results_stat 字典。统计过程分别由辅助函数 flush_queue 与 update_pair 函数执行。

在汇总数据的同时，主进程每隔一段时间就执行可视化函数，将当前比赛过程显示于控制台。可视化分为两个部分：比赛状态表格与选手排名。

比赛状态表格（Status）：

比赛状态表格内每个单元格横纵轴分别代表先手、后手参赛玩家，按玩家名称字典序排序，行列表头内容相同，只显示行表头。另外，每行末尾显示了每个玩家当前的得分情况。

表格对角线显示内容为“*****”，用于对照横纵坐标内容；显示 xx-yy 表示正在进行比赛，其中 xx 为该行玩家（先手）得分，yy 为该列玩家（后手）得分；若显示 results_stat 内符号说明多局比赛与统计已结束；若显示空白则说明未开始比赛。

比赛得分按行统计，其中“+”记 3 分，“0”记 1 分，其余不计分。

得分排名（Ranking）：

所有参赛玩家将按得分进行排序，并显示相应的柱状图；根据 MAX_PROMOTION 值将在相应排名处划线。

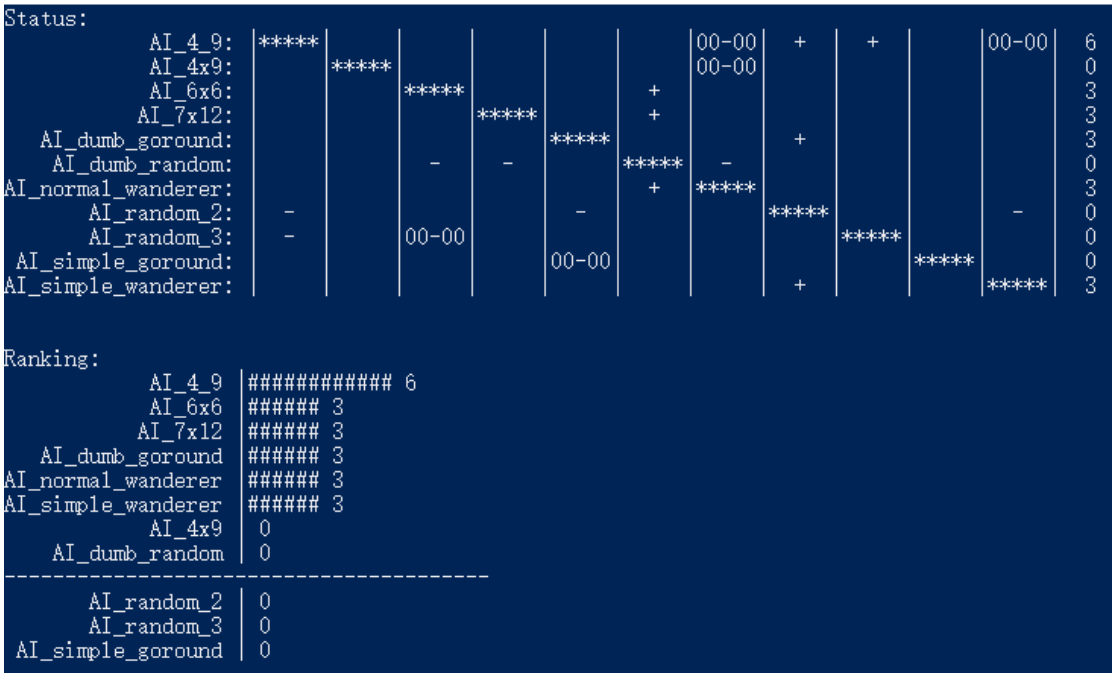

图 3 可视化界面示例

4.3. 数据库记录

循环赛主线程初始化时建立了一个简单的数据库，其中包含两个表单：match_raw 表记录了每局双方玩家名称、胜者编号（0 或 1，-1 表示平局）及胜利原因，match_result 表则记录了每个对手组合最终的胜负场数。依需求不同该部分可以扩充更多内容。

数据库建表命令：

a. 原始数据：CREATE TABLE `match_raw` (`plr1` TEXT, `plr2` TEXT, `winner` INTEGER, `type` INTEGER)

b. 统计结果：CREATE TABLE `match_result` (`plr1` TEXT, `plr2` TEXT, `flag` TEXT, `win1` INTEGER, `win2` INTEGER, `tie` INTEGER, PRIMARY KEY(`plr1`,`plr2`))