

郑州大学管理学院

本科毕业论文

题目：面向多纳什均衡的 Nashsweeper 设计及应用

学生姓名及学号	钱泽昊 201907070222
指导教师及职称	赵倩倩 讲师
专 业	工业工程
班 级	2019 级 1 班

2023 年 5 月 23 日

目 录

摘 要	1
Abstract	2
1 绪 论	3
1.1 Nash Equilibrium 背景介绍	3
1.2 国内外研究现状	4
1.3 小结	6
2 研究工作	7
2.1 研究路线	7
2.2 研究贡献	7
3 Nashsweeper 算法与数据库构建	9
3.1 纯纳什均衡的计算	9
3.2 SQLite 数据库设计	9
3.3 一种 Regret-base 算法的高效实现	10
4 Nashsweeper 系统设计与实现	12
4.1 系统架构	12
4.2 用户图形界面设计与实现	13
4.3 Nashsweeper 的其他技术实现	15
4.4 棋盘中每个单元格状态	18
4.5 游戏通关条件	19
4.6 游戏流程图	19
5 案例分析	20
5.1 输入数据	20
5.2 Regret Value 计算	21
5.3 初始化用户图形界面	21
5.4 开始游戏	22
6 结论及建议	25
6.1 Nashsweeper 的应用前景	25
6.2 Nashsweeper 的限制	25
6.3 Nashsweeper 的未来发展方向	25
致 谢	26
附录 A: Nashsweeper 核心计算引擎	27
附录 B: Nashsweeper 使用文档	30
参考文献	31

摘 要

纳什均衡是博弈论的核心概念，在很多领域都有基于 Nash Equilibrium 的应用，本篇毕业论文详细介绍了从基于 Regret Value 的纳什均衡求解算法实现到数据库设计，再从用户界面设计、系统设计到项目部署设计的一个被称为“Nashsweeper”的游戏，该项目旨在通过游戏寻找纯策略纳什均衡，让用户了解熟悉博弈论并掌握其基本理论。

除此之外，由于 Nashsweeper 的计算存储核心是基于 Python 以及 SQLite 的，因此我还开发了命令行界面的求解器，支持学术研究以及工业界相关人员在计算机程序以及 Jupyter Notebook 中调用 Nashsweeper 核心计算引擎，计算超大型博弈的所有纯纳什均衡。

本论文代码、使用文档以及部署方式已全部上传至 Gitee 代码仓库，可通过下面的链接进行访问：https://gitee.com/qian_zehao/nashsweeper

关键词：纯纳什均衡；非合作博弈；纯策略游戏；Nashsweeper。

Abstract

Nash Equilibrium is the core concept of game theory, and it has applications based on Nash Equilibrium in many fields. This graduation thesis introduces in detail from the realization of Nash Equilibrium algorithm based on Regret Value to database design, and then from user interface design, system design to A game called "Nashsweeper" was designed and deployed by the project. The project aims to find a pure strategy Nash equilibrium through the game, so that users can understand and be familiar with game theory and master its basic theory.

In addition, since the computing and storage core of Nashsweeper is based on Python and SQLite, I have also developed a solver for the command line interface to support academic research and relevant personnel in the industry to call the Nashsweeper core computing engine in computer programs and Jupyter Notebooks , computing all pure Nash equilibria of very large games.

The code, usage documents and deployment methods of this thesis have all been uploaded to the Gitee code repository, which can be accessed through the following link:

https://gitee.com/qian_zehao/nashsweeper

Keywords: Pure Nash Equilibrium; Non-cooperative Game; Pure Strategy Game; Nashsweeper.

1 绪 论

1.1 Nash Equilibrium 背景介绍

纳什均衡是博弈论中的一个重要概念，它指的是在一个博弈中，当所有参与者都不改变自己的策略时，没有人能够通过改变策略而获得更多的收益。也就是说，每个参与者的策略都是对其他参与者策略的最佳反应。

纳什均衡的概念是由美国数学家约翰·纳什^[1]在 1950 年提出的，他在博士论文中证明了在有限参与者和有限策略的博弈中，至少存在一个纳什均衡。1994 年，他因为在非合作博弈的均衡分析方面做出了开创性的贡献，而与另外两位博弈论学家共同获得了诺贝尔经济学奖。纳什的人生故事也被改编成了电影《美丽心灵》。

纳什均衡在管理科学与工程^[2]、工业工程^[3]、经济学、计算机科学、演化生物学、人工智能、会计学、政策和军事理论等领域都有广泛的应用。一些经典的纳什均衡案例包括囚徒困境、智猪博弈、普通范式博弈等。这些案例展示了在不同的博弈情境下，参与者如何根据理性和利益来选择最优的策略，以及这些策略可能导致的结果。

1.1.1 Nash Equilibrium 的基本数学模型

参与者、每个参与者的策略集、各参与者的收益函数,是博弈论最重要的基本要素。

非合作博弈的基本要素 $\Gamma(N, S, u_i)$:

- (1) N 代表一组有限的玩家, i 为索引, $|N| = n$;
- (2) S 表示每个玩家 i 的动作集元组, $S = (s_1, s_2, \dots, s_n)$;
- (3) $s \in S$ 是一个动作, S_i 表示玩家 i 的策略集;
- (4) u_i 表示收益函数。

Game(游戏): 在泛化的场景中, 游戏包含了一些参与者、行为和策略以及最终收益。

Players(参与者): 是指参与到游戏中的具有理智的实体。

Payoff(收益): 收益也可以称为奖励, 是玩家在参与游戏中会得到一个结果, 可以为正也可以为负。

纳什均衡是指在一个博弈过程中, 无论对方的策略选择如何, 当事人一方都会选择某个确定的策略, 则该策略被称作支配性策略。如果任意一位参与者在其他所有参与者的策略不变的情况下改变自己的策略, 都不能使自己获得更好的收益, 则称该策略组合为纳什均衡。

简单来说, 纳什均衡是指在博弈中, 每个人都采取了最优策略, 没有人可以通过改变自己的策略来获得更多的收益。

因此, 达到纳什均衡时的条件为: 对所有 $i \in N$ 和 $s' \in S$, 有 $u_i(s_i^*, s_{-i}^*) \leq u_i(s_i', s_{-i}^*)$

1.1.2 纯纳什均衡求解器

用于寻找在众多策略中纳什均衡点的计算机程序, 常用解法有数学规划、机器学习、强化学习以及元启发算法。本论文使用的是 Regret-base 算法在大规模策略中寻找纳什均衡解。

1.1.3 纳什均衡在工业工程领域的应用

纳什均衡在工业工程中的应用有很多, 例如:

- 在**供应链管理**中, 纳什均衡可以用来分析供应链成员之间的博弈行为, 如定价、库存、订货、合同等, 以及如何设计合作机制来提高供应链的整体效益。
- 在**生产计划和调度**中, 纳什均衡可以用来分析多个生产者之间的竞争和协作, 如如何分配资源、优化生产顺序、协调生产节奏等, 以及如何通过激励机制来促进生产者之间的合作。

• 在**质量管理和可靠性工程**中，纳什均衡可以用来分析质量保证体系中的各个主体之间的博弈行为，如：如何制定质量标准、质量检验、质量奖惩等，以及如何通过信誉机制来维护质量信用。

1.1.4 纳什均衡的推广

鉴于纳什均衡在学术研究和工业生产领域的重要影响地位，关于纳什均衡理论介绍主要集中在数学、运筹学以及经管类书籍，在建模求解方面内容相对晦涩且对读者有一定入门门槛；海尔、富士康等制造业企业也有关于内部管理人员的培训，专门讲授用于生产管理博弈论。

运筹 OR 帷幄等国内的专门推广运筹学的机构，其中很多知乎文章、问题回答和视频关于纳什均衡解决实际工业问题。同时，国外的高校也在开发基于纳什均衡理论的游戏，试图通过精致美观的图形界面帮助用户以及其他用户从 0 基础开始入门博弈论理论。

本篇论文的核心主题 Nashsweeper 也是一个从纯纳什均衡求解器的实现到数据库设计，再从用户界面设计到项目部署的游戏，该项目旨在通过游戏寻找纯策略纳什均衡，让用户了解熟悉博弈论并掌握其基本理论。

1.2 国内外研究现状

本部分将通过文献综述从纳什均衡求解器研究发展，介绍基于数学优化、元启发式算法以及机器学习三个领域的求解器，以及用于纳什均衡推广的游戏和计算机程序。

1.2.1 纳什均衡求解器研究

纳什均衡问题（NEP， Nash Equilibrium Problem）由于是 NP-hard 问题，并没有精确解法，只有在特殊情况下才有可能求出精确解法。这是因为在一些情况下，博弈中的参与者会有多种选择，而这些选择会相互影响，导致很难求出精确解，并在一定程度上限制了博弈理论的普遍应用。下面（表 1.1）将从元启发算法、机器学习算法以及数学优化算法论述近代纳什均衡求解器研究：

表 1.1：代表性论文研究

	算法名称	年份	优势	劣势
数学优化	指数型惩罚函数方法 ^[4]	2015	在特定约束情况下，可以得到部分精确解	需要约束条件多，使用到商业求解器，无法应对多维、非凸的目标函数
	列约束生成算法 ^[5]	2020		
	拟变分不等式+正则化函数+Gap 函数 ^[6]	2020		
元启发	改进麻雀算法 ^[7]	2022	相较数学优化算法，在求解多目标博弈求解速度更快	只能得到解析解，部分算法会陷于局部收敛
	博弈人工蜂群算法 ^[8]	2023		
	微分进化算法 ^[9]	2023		
	协同进化 ^[10]	2020		
机器学习	强化学习 ^[11]	2019	泛化能力强，适合多智能体博弈	求解过程复杂，强化学习建模困难
	贝叶斯纳什均衡 ^[12]	2022		

A. 基于数学优化算法的纳什均衡求解器

基于传统的数学优化算法求解纳什均衡问题有数学规划法（1999）、解析法（2003）、列约束生成算法（2023）等，其核心思路就是通过约束条件，将纳什均衡模型转化成凸优化问题，在特定点可以得到纳什均衡。如：侯剑^[13]（2013）和陈盼华（2020），都是基于 NEP 问题特性与 QVI 关系，将问题转化为拟变分不等式问题进行建模，并通过 Nikaido-Isoda 正则化函数、gap 函数将问题转化为光滑无约束最优化问题求解。

使用数学优化的优势在于面对较低维度的纳什均衡问题可以快速得到精确解，但是需要对纳什均衡问题进行转化，如许吉祥（2015）通过证明若光滑的惩罚 NEP 序列的解序列的聚点处 EMFCQ 成立，那么此聚点是 GNEP 的解，类似的，数学优化方法需要加入过多的约束条件才能成立。

同时，在处理大规模线性规划时需要专门安装商业或是开源的求解器，如：CPLEX, Gurobi, Mosek, LINGO 和 SCIP 等，这些求解器安装调用复杂，不利于初学者入门使用。

鉴于数学优化算法解决 NEP 问题对于模型约束条件和目标函数的严苛条件（凸函数、光滑、存在特定点等），以及求解器的复杂使用方法，近年来流行基于元启发算法以及机器学习算法计算纳什均衡解析解。

B. 基于元启发式算法的纳什均衡求解器

周艳杰和 Kim (2020) 提出了一种协同进化求解程序来计算纯纳什均衡，该算法可以计算所有纳什均衡，然而，协同进化求解过程非常复杂，对初学者来说不是很友好。

使用元启发算法（如：蚁群算法、人工蜂群算法等）适合处理 NEP 问题，因其操作简单，且算法复杂度低、求解效率高。李慧敏^[14]（2022）通过自适应差分粒子群算法进行单目标博弈的求解。

张国强（2023）通过使用微分进化算法，在所有策略中随机选取三个不同位置的亲代个体进行变异和交叉操作产生新的子代个体，保留优势策略，淘汰劣势策略，使用微分进化算法既保持了种群多样性，又增强了算法的全局寻优能力。

群体智能算法很好的可以处理多维，非凸性等复杂的目标函数，在处理 NEP 问题上具有一定的广泛适用性。

C. 基于机器学习算法的纳什均衡求解器

由于机器学习类算法是非模型依赖的，因此近年来有很多基于机器学习求解纳什均衡解的研究。

赵倩（2022）基于 Q-learning 的多智能体系统最优一致性的研究中基于强化学习^[15]Q-learning 算法以及最小二乘法多智能体控制系统解析解。李佳莲^[16]（2023）在样本高效的强化学习方法研究中，基于两人完全的零和扩展型博弈模型，设计了通过强对手策略进行学习的算法。

1.2.2 用于教学的博弈论游戏开发

博弈论是经济学、工业工程、系统工程、计算机科学、数学、管理科学等专业本科生和研究生的必修课。甚至美国康奈尔大学数学系开始将博弈论引入中学和高中的教学课程 (Anema^[17] 2008)。

为了面向公众而非数学、经济管理、工程类专业人士普及博弈论相关理论知识，学术界设计了很多基于纳什均衡的实验和游戏，通过和用户间的交互潜移默化传授相关内容。Fan^[18](2000) 利用囚徒困境游戏来探索儿童的行为，以教授儿童合作博弈。Fan(2000) 将囚徒困境环境与纸牌游戏相结合，并招募了台北民春小学的孩子们进行实验。不足之处在于 Fan 手工设计的纸牌游戏工作量非常大，手工分析统计结果的效率低。Rothman^[19] (2012) 采用国际谈判来教授应用博弈论。为提升经济和博弈论这两门学科的教学质量，Ghic 和 Grigorescu^{[20][21]}在 2014 年开发了下面（图 1.1）这个经济博弈论游戏。

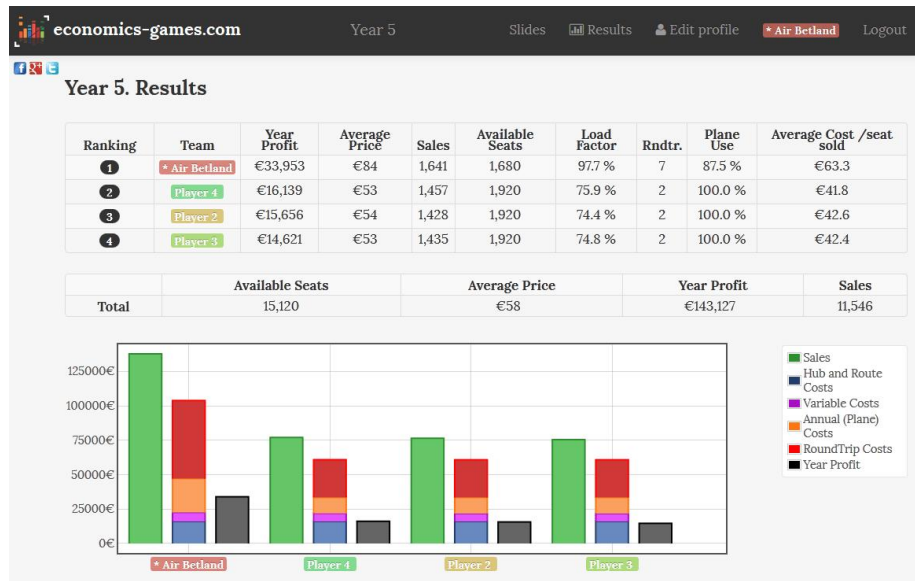


图 1.1: 经济学游戏 (源: economics-game.com)

Badeau^[22]等人 (2014) 提出, 将数字教育游戏引入课堂教学将导致用户的积极参与, 他们将经济学游戏提供的游戏作为案例研究用于教授实验经济学课程。O'Roark 和 Grant^[23] (2018) 将漫画书中的场景转化为特定的游戏形式, 以帮助用户培养寻找和解释游戏解决方案的技能。Abasian^[24]等人 (2020) 开发了一个在线教育游戏来说明运输游戏。Iranpoor^[25] (2021) 引入了 Knights Exchange Puzzle 来教用户模型的效率。

1.3 小结

在 1.2.1 部分中通过对国内外现有的纳什均衡求解器对比研究后发现, 现有基于数学优化与机器学习算法 NEP 求解器使用繁琐且对问题模型约束条件多, 对于博弈论入门者的前置基础知识储备要求高, 而基于元启发算法的 NEP 求解器虽然容易实现, 但是不一定可以求解全部纳什均衡点, 因此在本文中提出的 Regret-base 算法可以精确求解大规模策略的全部纳什均衡。

通过文献综述, 到目前目前还没有任何关于通过玩互动博弈来计算纯纳什均衡的已发表作品。因此在本文中, 我们开发了一个有趣的游戏来帮助用户直观地理解纯纳什均衡的计算过程。

2 研究工作

2.1 研究路线

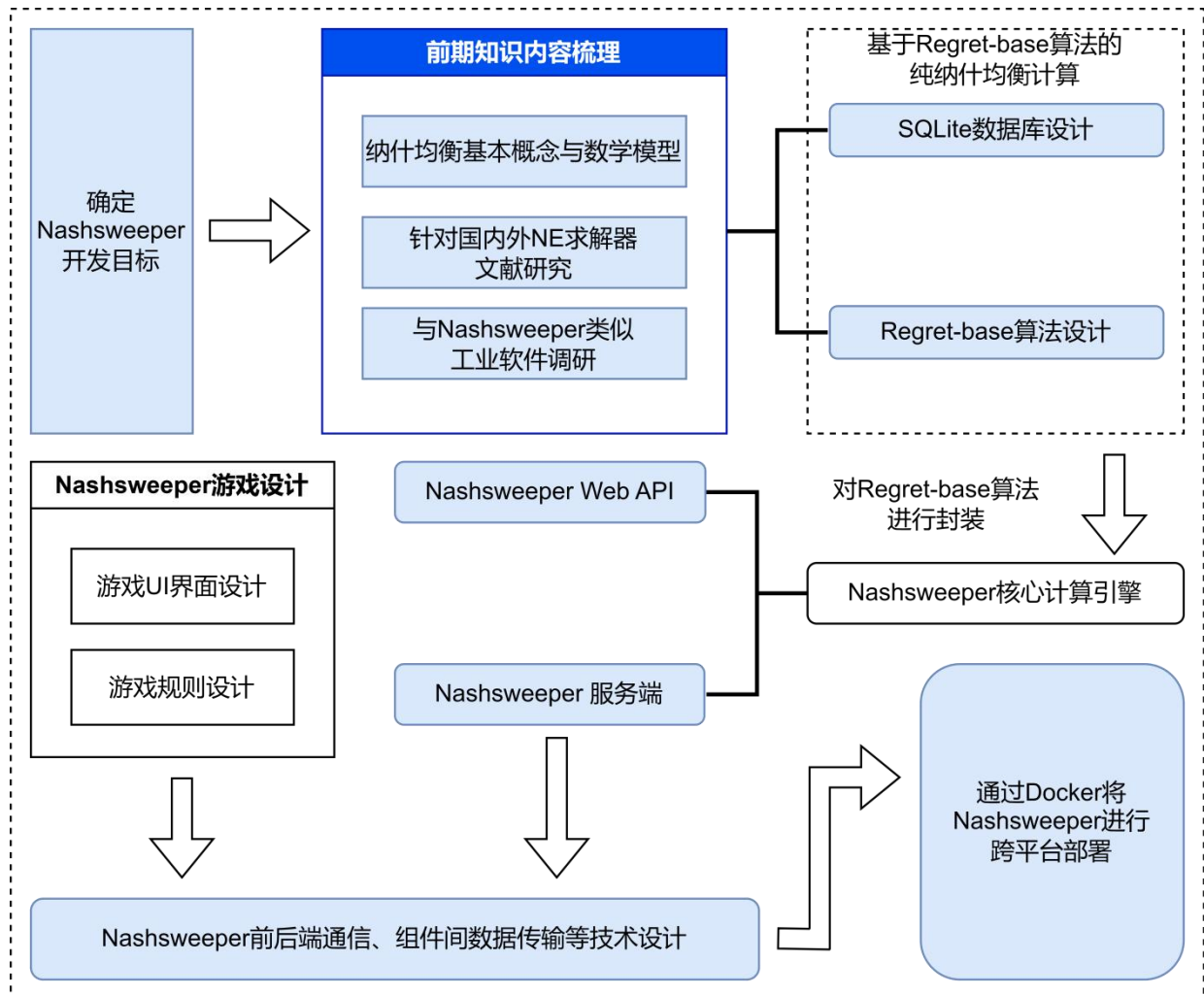


图 2.1: Nashsweeper 研究流程图

2.2 研究贡献

2.2.1 论文部分

- a. Nashsweeper-core: 基于 Regret-value 算法开发计算全部纯纳什均衡精确解的求解器;
- b. 本文基于 Nashsweeper-core 提出了一个名为 Nashsweeper 的免费教育游戏, 论文第四部分有对 Nashsweeper 系统设计与实现的详细介绍, Nashsweeper 用于通过计算纯策略纳什均衡来教授博弈论。Nashsweeper 有很多特点, 包括开源代码, 用户界面友好, 无需安装, 跨平台等。Nashsweeper 是一款交互式单人益智视频游戏, 类似于流行的 Minesweeper (Kaye 2017)。

2.2.2 代码部分

- a. 基于渐进式前端框架 Vue.js, 后端开发 Python+Flask+NumPy+Pandas, SQLite 数据库, 前端样式语言 TailwindCSS, 组建间通信库 Pinia, 前后端异步通信 Axios 等开发的专为学习博弈论的初学者而设计的跨平台游戏——Nashsweeper;
- b. 本文还开发了 Nashsweeper 的命令行界面 (CLI) 版本, 基于 Nashsweeper-core-engine, 以支持计算纯纳什均衡的超大规模非合作博弈。Nashsweeper 的

命令行界面版本采用 Python 编程语言开发，集成了 SQLite 数据库，Nashsweeper 的命令行界面版本专为需要库来计算大型非合作博弈的纯纳什均衡的研究人员而设计。

c.为了方便用户在自己的设备上部署 Nashsweeper，我编写了用于运行前端 Vue 服务，以及运行 Python 后端服务的 dockerfile，并通过 docker-compose 实现在 Linux 或是 Windows WSL 操作系统下快速启动部署两个应用容器。

2.2.3 用于学术以及技术社区交流

a.为了便于项目源代码管理、软件更新迭代以及用户自定义功能和游戏规则编写，本项目的全部源代码已经上传至国内开源平台 Gitee，可通过 git clone 从项目仓库克隆，项目地址：https://gitee.com/qian_zehao/nashsweeper。

b.Nashsweeper 技术博客：使用 Wordpress 搭建的 Nashsweeper 官方网站，上面记录了 Nashsweeper 的开发流程，使用文档以及技术细节，便于用户查看使用。项目官网地址为：<http://www.nashsweeper.openie.xyz:803>

c.针对加入 Nashsweeper Gitee 仓库成为本项目的贡献者、程序中的 Bug 以及 Nashsweeper 改进意见，用户可通过 Email：qianzehao123@gmail.com 联系作者。

2.2.4 开发以及运行 Nashsweeper 项目

Nashsweeper 在分为开发模式以及生产模式（图 2.2），在开发模式下需要安装一些依赖（Nodejs \geq 18.10, Miniconda \geq 3.9, Docker）（推荐在 Ubuntu 22.10 下进行开发）。更多技术细节详见 https://gitee.com/qian_zehao/nashsweeper 网站。

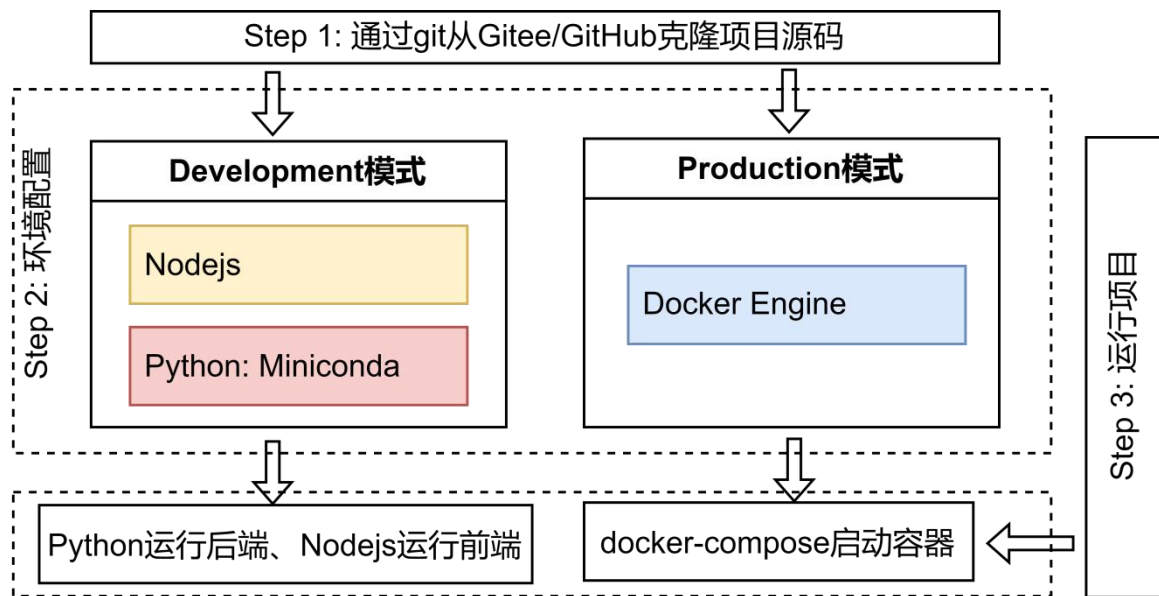


图 2.2: Nashsweeper 开发/部署模式图

3 Nashsweeper 算法与数据库构建

本部分将基于 Regret-base 的针对非合作博弈的纯纳什均衡求解算法以及配套使用的 SQLite 数据库存储设计进行讨论。

3.1 纯纳什均衡的计算

根据第一部分绪论中的面向非合作博弈的纯策略纳什均衡的定义，我们给出非合作博弈的数学定义，一个有限的， n 个参与者的游戏可表示为 $\Gamma(N, S, u_i)$ ，其中：

- N 代表一个有限集合的参与者，索引为 i ， $|N| = n$;
- S 代表参与者 i 的动作集合元组， $S = (s_1, s_2, \dots, s_n)$ ， $s \in S$ 是动作集合， S_i 代表参与者 i 的策略集合。
- u_i 是参与者 i 的效用函数， $u_i: s_1 \times \dots \times s_n \rightarrow R, u = (u_1, \dots, u_n)$ 。

对于给定的 N 个参与者的博弈 $\Gamma(N, S, u_i)$ ，对于任意 $i \in N$ ， $s'_i \in S_i$ ，当如下不等式(1)成立时，策略 (s_1^*, \dots, s_n^*) 是一个纯纳什均衡：

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s'_i, s_{-i}^*) \quad (1)$$

到目前为止还没有很多用于计算纯纳什均衡精确解的算法，Wu^[26] (2014) 等人提出了一种基于 0-1 规划计算纯策略纳什均衡的方法。通过使用开源或是商用的数学规划求解器，例如：CPLEX, Groubi, Mosek, Lingo 等，可以获取所有纯纳什均衡。而本文试图去设计一个轻量化框架去计算纯纳什均衡，我们不希望去使用混合整数规划求解器去增加用户使用负担。因为这些混合整数规划以及 0-1 规划数学模型对初学者并不友好，另一个原因是第三方轻量化求解器对于研究人员来说更能接受，因为这样可以降低代码配置的复杂程度。因此在本文中，我们采纳了近年来由 Corley^[27] (2020) 等人提出的 regret-based 算法，它是一个枚举算法，核心是 regret 函数 r_i ，表示如下：

$$r_i(s) = \max_{s'_i \in S_i} (u_i(s'_i, s_{-i}) - u_i(s_i, s_{-i})) \quad (2)$$

通过使用 regret 函数，Corley 等人介绍了一种纳什均衡的替换定义，如下引理所示：

定理 1: 当对于所有 $i \in N$ 时， $r_i(s) = 0$ ，则纯策略 s^* 是博弈 $\Gamma(N, S, u_i)$ 的纳什均衡。

另一个重要概念是最佳响应 (Best Response)，定义如下：

定义 1: 如果存在一个参与者 i 且 $r_i(s) = 0$ ，则纯策略 s 对于博弈 $\Gamma(N, S, u_i)$ 是一个纳什均衡。

基于上述定理 1 和定义 1，本论文提出如下引理：

引理：当策略 s 对于所有参与者 $i \in N$ 都是 Best Response，则称对于博弈 $\Gamma(N, S, u_i)$ ，纯策略 s^* 为其纳什均衡。

由于引理很容易结合 Best Response 的定义和定理 1 证明，因此我们省略对其的证明。

3.2 SQLite 数据库设计

正如之前所提到的，参与者 i 拥有策略集 S_i ， $|S_i|$ 代表 s_i 的基数，然后我们可以很容易得到参与者 i 的纯策略总数，如下所示：

$$TS_i = \prod_{i=1}^{|N|} |S_i| \quad (3)$$

TS_i 代表参与者 i 的纯策略总数，所有的参与者的纯策略总数可以用 TS 表示， TS 表示如下：

$$TS = |N| \prod_{i=1}^{|N|} |S_i| \quad (4)$$

通过表示 TS 的等式，不难发现 TS 是一个指数增长型函数，随着 N 和 S_i 的增长， TS 将会变成一个特别大的整数，这将导致所有的策略无法被完全加载到计算机的内存里面。McKelvey^[28] (2006) 等人开发的 Gambit 是一个流行的工具用来计算博弈论，但是它不支持分析大型博弈。为了支持大规模非合作博弈，在本文中，我使用到了 SQLite 来存储策略。SQLite 是一个小巧、轻量且快速的 SQL 数据库引擎。对于任意的 n 元组的策略，它会有相应的 n 元组回报函数的 (payoff function)。因此，我们将有一个 $2n$ 元组用来记录策略和回报函数。总的来说，我们将在 SQLite 数据库中有 $\prod_{i=1}^{|N|} |S_i|$ 条记录。从等式(2)中，当通过给定其他参与者的策略 s_{-i} 去计算参与者 i 的 regret value 时，我们需要去计算最大的 u_i ，为了使寻找所有的 (s'_i, s_{-i}) ，我们需要添加独特的 ID 信息去帮助寻找 s'_i 。给定一个动作 (s_1, s_2, \dots, s_n) ，每个 s_i 被连接起来生成一个字符串作为唯一 ID，可以很容易地用于计算 Regret Value。下方表格 3.1 展示了策略和回报值的数据库：

表 3.1：策略和回报值的数据库

ID (string)	Strategies(string)			Payoff value(real number)		
	Player 1	...	Player n	Player 1	...	Player n
$s_1^1 - \dots - s_n^1$	s_1^1	...	s_n^1	$u_1(s_1^1, \dots, s_n^1)$...	$u_n(s_1^1, \dots, s_n^1)$
...
$s_1^{ S_1 } - \dots - s_n^1$	$s_1^{ S_1 }$...	s_n^1	$u_1(s_1^{ S_1 }, \dots, s_n^1)$...	$u_1(s_1^{ S_1 }, \dots, s_n^1)$
...
$s_1^{ S_1 } - \dots - s_n^{ N }$	$s_1^{ S_1 }$...	$s_n^{ N }$	$u_1(s_1^{ S_1 }, \dots, s_n^{ N })$...	$u_1(s_1^{ S_1 }, \dots, s_n^{ N })$

3.3 一种 Regret-base 算法的高效实现

接下来，我将简要概括基于 regret value 算法的纯纳什均衡的设计与实现。Corley 等人所提出的基于 regret value 算法并且分析其计算复杂度时没有考虑到等式 (2) 中的 $\max_{s_i \in S_i} (s'_i, s_{-i})$ 。正如我们上一小结所提及的数据库中有 $\prod_{i=1}^{|N|} |S_i|$ 条记录，算法复杂度为 $o(n^2)$ ，这将导致计算 $\max_{s_i \in S_i} (s'_i, s_{-i})$ 将非常费时。因此我们基于 SQLite 设计了算法，如 Algorithm 1 所示：

Algorithm 1: Regret-based algorithm with SQLite.

Input: SQLiteDataBase, N

Output: NEset, BRset

```

1.  for  $i \in N$  do                                // for loop of each player
2.      StrategySet = PermutationsFunction( $N \setminus i$ );
3.      for  $ss_j \in \text{StrategySet}$  do
4.          MaxPayoff $_i = 0$  ;                      // the maximum payoff value of player i
5.          while SELECT ( $s_i', ss_j$ ) FROM SQLiteDataBase do
6.              if  $u_i(s_i', ss_j) > \text{MaxPayoff}_i$  then
7.                  MaxPayoff $_i = u_i(s_i', ss_j)$ ;
8.              end
9.          end
10.          $r_i(s_i', ss_j) = \text{MaxPayoff}_i - u_i(s_i', ss_j)$ ;
11.     end
12. end
13. StrategySet = PermutationsFunction(N);
14. for  $s \in \text{StrategySet}$  do
15.     NashFlag = false;
16.     BestResponseFlag = false ;
17.     for  $i \in N$  do
18.         if  $r_i(s) = 0$  then
19.             NashFlag = NashFlag AND true ;
20.             BestResponseFlag = BestResponseFlag OR true ;
21.         end
22.     end
23.     if NashFlag then
24.         NEset = NEset  $\cup$  s;
25.     else if BestResponseFlag then
26.         BRset = BRset  $\cup$  s;
27.     end
28. return NEset, BRset;
```

4 Nashsweeper 系统设计与实现

虽然非合作博弈至少需要两个玩家来玩。Nashsweeper 的一个目的是验证用户是否可以在给定的非合作博弈中找到纳什均衡。因此，在本文中，Nashsweeper 被计为一个单人游戏，用于寻找所有纯纳什均衡。

4.1 系统架构

Nashsweeper 游戏专为用户和学术研究人员设计，用于学习博弈论或计算纯纳什均衡。Nashsweeper 游戏需要满足跨平台标准，客户端可以在 Windows、Linux 和 MacOS 系统上运行。便携是 Nashsweeper 游戏需要满足的另一个标准。凭借便携功能，Nashsweeper 游戏无需安装或卸载。为了满足上述标准，我们设计了两个版本的 Nashsweeper 游戏。带有 GUI 的 Nashsweeper 游戏是一个基于 Vue^[29]和 JavaScript 以及 Python Flask 后端开发的基于 Web 的应用程序。带有 CLI 的 Nashsweeper 游戏是在 Python 编程语言和 SQLite 下开发的控制台应用程序以及 Nashsweeper Web API 网络后台接口。

4.1.1 基于浏览器的跨平台客户端

传统安装类应用程序如运行在 PC、MacOS 等操作系统基于 JavaScript 语言的 Electron、Tauri 框架的桌面端应用程序，以及近 5 年来热门的在手机、Pad 等移动平台上的 React Native、Flutter，这些程序框架能更好的调用硬件设备（Wi-Fi、蓝牙、陀螺仪、加速度计等）进行交互，但是这些框架一个无法忽视的缺点是需要安装。而 Nashsweeper 作为一个旨在面向用户普及博弈论和纳什均衡理论，以及推广基于 Regret-base 算法的纳什均衡求解器的游戏，过于复杂的安装方式会丧失很多用户，Nashsweeper 客户端在最初的技术选型的时候为了跨平台开发以及避免使用到过多编程语言，因此采用了浏览器访问的模式，下图（图 4.1）为 Nashsweeper 客户端架构：

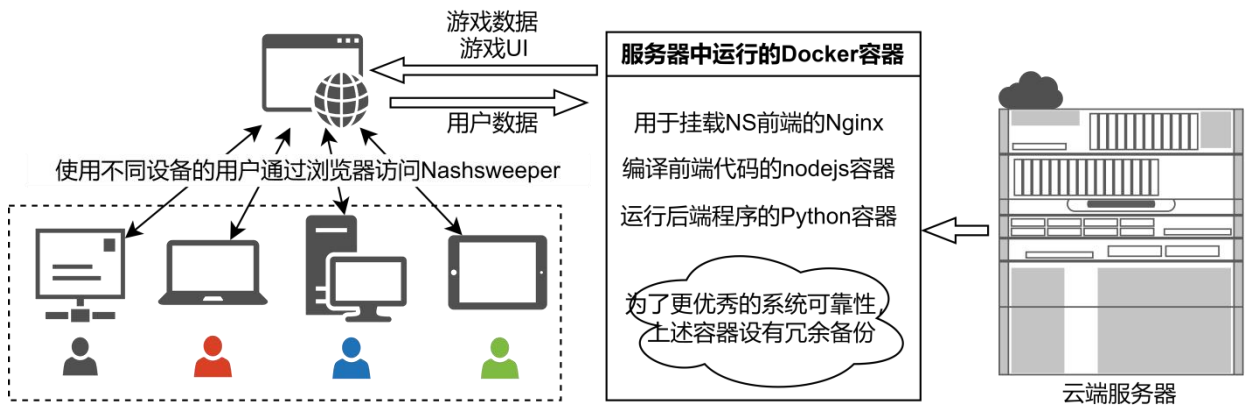


图 4.1: Nashsweeper 客户端架构

4.1.2 基于 Linux + Docker 跨平台服务端

虽然不像前端用户交互界面跨平台那样需要考虑大量的问题，但是 Nashsweeper 服务端跨平台同样也在我的设计范围之内，Nashsweeper 在服务端需要使用到 Python、担任后端的 Flask、处理用户上传游戏数据的 *.csv 格式文件所用到的 Pandas^[30]、进行 regret-base 算法计算中调用的 NumPy^[31]库等，若直接在 Linux 服务器上面直接运行 Nashsweeper 后端，可能会由于 Python 依赖冲突导致无法正常运行，同时考虑到一些用户出于隐私考虑，倾向于将 Nashsweeper 私有化部署在自己的设备上，因此 Nashsweeper 采用 Docker^[32]进行前后端服务的部署，同时使用 Docker 也可以支持 Nashsweeper 在不同的操作系统（MacOS、

PC 等)、不同的硬件平台 (x86、ARM、RISC-V 芯片) 上运行。下图为 Docker 基础设施下 Nashsweeper 服务端 (图 4.2) 架构:

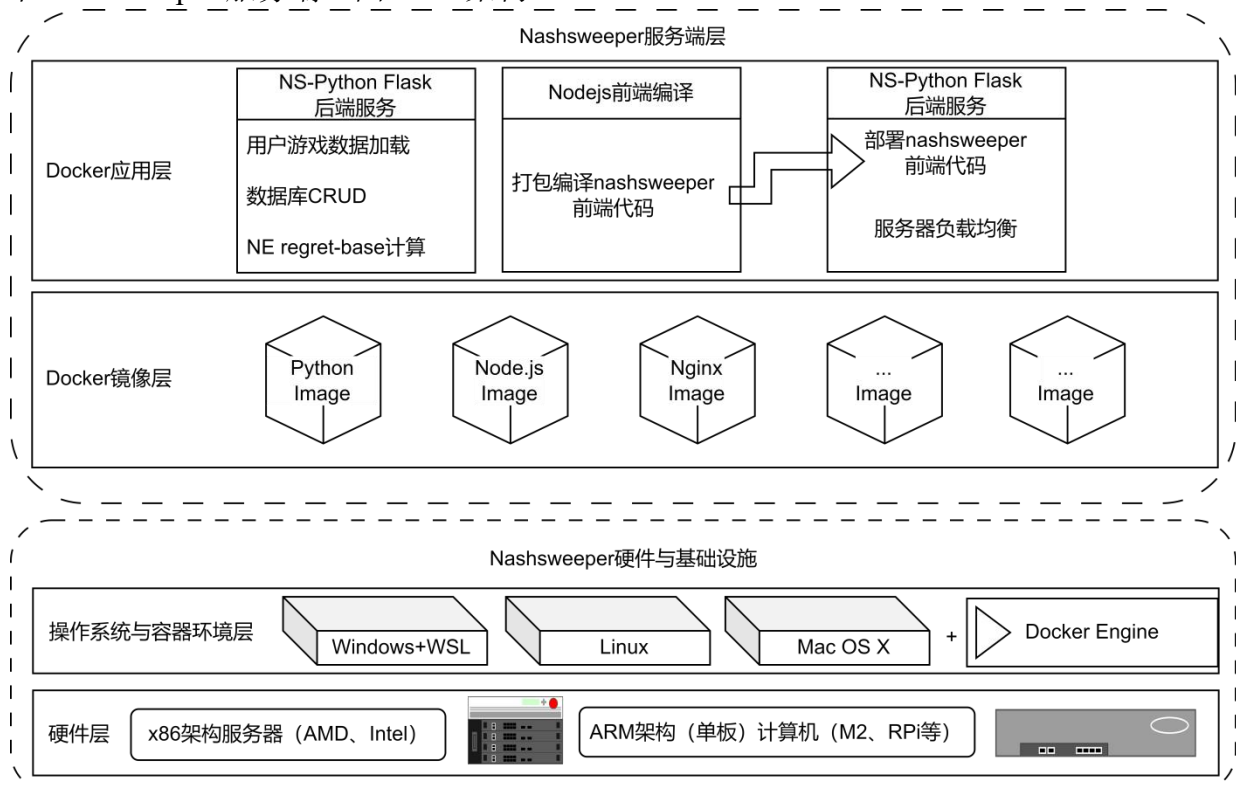


图 4.2: Nashsweeper 服务端架构

4.2 用户图形界面设计与实现

对于图形用户界面设计与开发是一场永无止境的战斗。清晰有序的游戏场景会更吸引用户玩 Nashsweeper, 给用户带来极佳的视觉体验。Nashsweeper 游戏的图形用户界面也是受与 Windows 系统集成超过 20 年, 目前仍然是 Windows Store 免费游戏前 5 名的扫雷游戏 Microsoft Minesweeper^{[33][34]}的启发, 其图形用户界面也是一个棋盘, 每个单元格显示策略及其相应的收益值。

下图 (图 4.3) 所示为 Nashsweeper 的图形用户界面原型设计, 图纸中的 A 部分显示着游戏信息, B 部分显示的是游戏用户数据如: 游戏耗时、最佳响应计数器 (Best Response Counter)、纳什均衡计数器 (Nash Equilibrium Counter)。图中以棋盘的形式展示了两玩家非合作博弈所有策略时的回报数值 (Payoff Value)。棋盘中的每一横行代表玩家 1 执行的策略, 每一纵列代表玩家 2 执行的策略。棋盘中每一个单元格 (cell) 中的一维数组 [payoff(x), payoff(y)] 代表当玩家 1 执行 x 策略, 玩家 2 执行 y 策略时, 各自的 payoff。C 部分的下拉数据表、以及右侧的三个染色表格代表游戏操作记录 (Log)。D 部分为本项目的开源代码链接以及全栈依赖库的官网, 下方的数据表格为玩家排名。

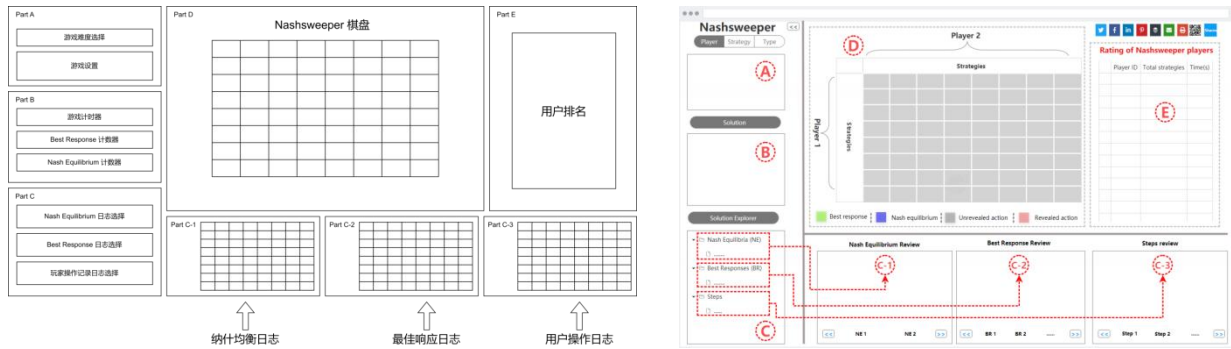


图 4.3: GUI 用户图形界面设计草图 (左) 与快速原型设计 (右)

4.2.1 通过*.vue 文件定义 Nashsweeper 功能组件

正如上文中的 Nashsweeper 图形界面设计图所示, 游戏界面包含很多模块: 用户信息、游戏面板、玩家排名、模式选择、计数&计时器模块等, 若采用传统 Web 前端开发 (HTML+CSS+JS), 则非常不利于维护, 因此在 Nashsweeper 中我们使用 Vue 定义的前端, 具体前端模块定义 (图 4.4) 以及 Nashsweeper 组件 Vue 标准模板 (Code 1) 如下所示:

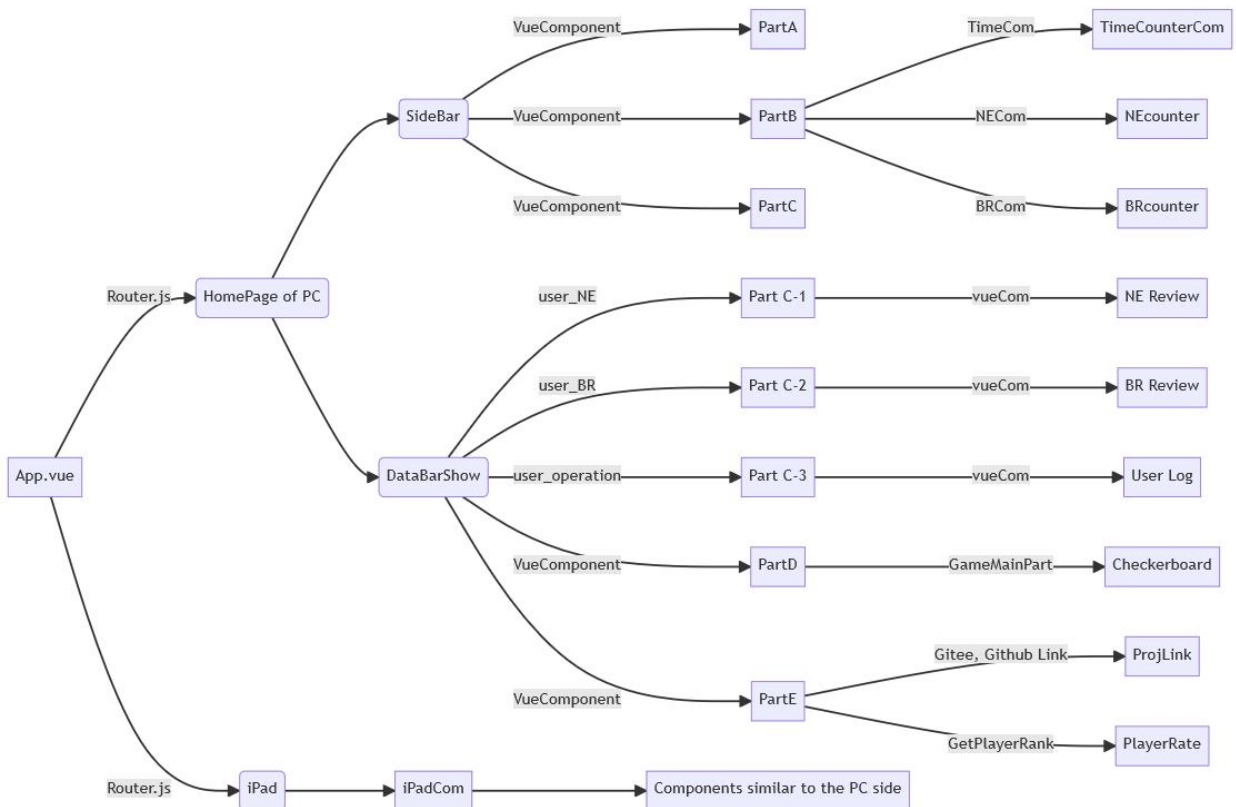


图 4.4: Nashsweeper 前端组件设计

Code 1: Nashsweeper 组件 Vue 标准格式

```

1. <template>
2.   <div>
3.     这个部分用于绘制 Nashsweeper 组件（button、checkerboard、link 等）
4.   </div>
5. </template>
6.
7. <script>
8.   import NPM 依赖或是自定义组件、js 脚本
9.   export default {
10.     name: "Nashsweeper 组件名称",
11.     components: {导入的组件需要在此实例化},
12.     data() {
13.       return {Nashsweeper 组件中的数据以及其他模块获取的数据}
14.     },
15.     methods: {定义方法，如计时器函数，可在 template 和 mounted 中被调用},
16.     mounted() {挂载 methods 中以及其他导入的 js 脚本，在组件被加载时执行},
17.   }
18. </script>
19.
20. <style scoped> Nashsweeper 组件样式定义
21.   这个位置用于定义组件的样式，使用 scoped 标签不会干扰其他组件的样式
22. </style>

```

4.2.2 通过 TailwindCSS 和 DaisyUI 控制页面布局

为了让 Nashsweeper 前端用户界面更美观以及更易于开发，使用到了“原子化”CSS 预处理引擎 TailwindCSS 编写前端样式，以及基于 TailwindCSS（图 4.5）开发的 DaisyUI（图 4.6）组件库。

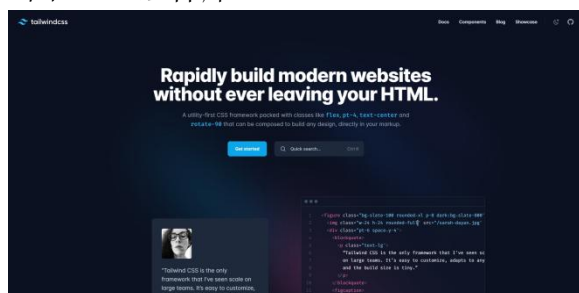


图 4.5: TailwindCSS

[Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.](https://tailwindcss.com/)

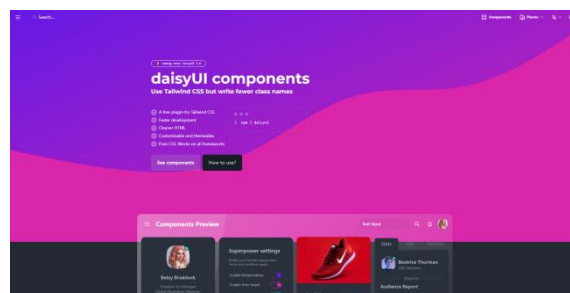


图 4.6: DaisyUI

[daisyUI — Tailwind CSS Components](https://daisyui.com/)

4.3 Nashsweeper 的其他技术实现

4.3.1 Nashsweeper-core-engine 核心计算引擎（Nashsweeper CLI）

在计算 Regret-value 时所要用到数据库，基本上所有数据库都需要专门驱动程序或是代码才能实现对数据库的 CRUD 操作，包括 SQLite，例如 Node.js 在读写 SQLite 时需要依赖 node-gyp 编译 C 的 SQLite 驱动，这对于 Docker 部署的 Nashsweeper 后端非常不方便。

Python 3 开始 Python 解释器自带 SQLite 数据库及其驱动程序让 Nashsweeper-core-engine 实现变得简单。下图展示了从上传文件夹获取到的用户 Nash 均衡配置文件读取到使用 NumPy 计算 Regret-value 到最终计算输出 NE 结果的 Nashsweeper-core-engine 程序运行流程（图 4.7）：

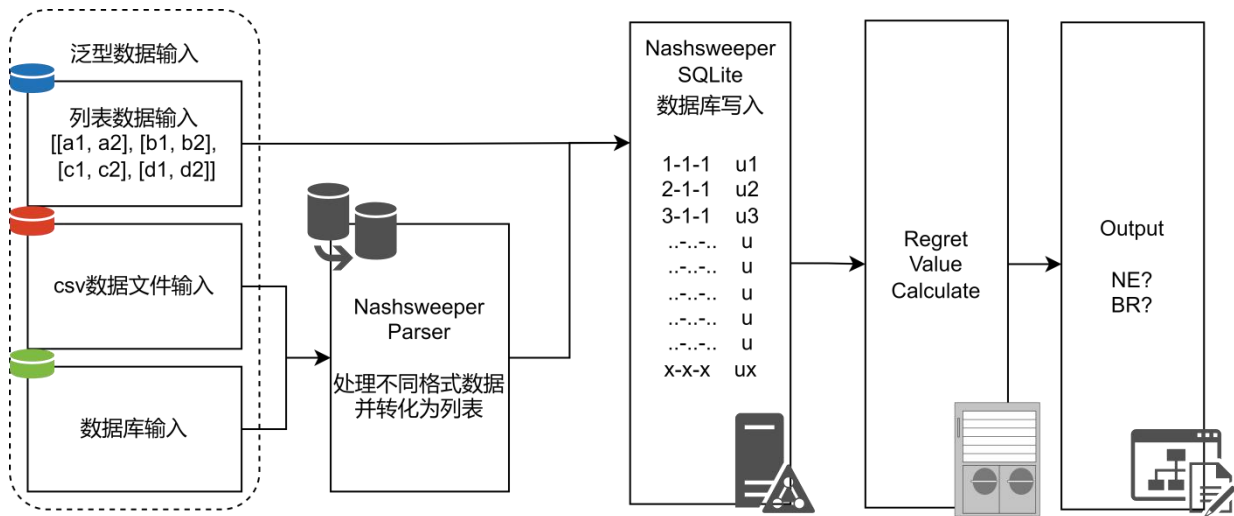


图 4.7: Nashsweeper-core-engine 数据处理流程

4.3.2 Nashsweeper 后端开发及 Nashsweeper Web API 实现

受 ChatGPT 母公司 OpenAI 对外开源的 OpenAI API 的启发，Nashsweeper 也通过 API 的方式对外提供服务，用于计算超大规模纯策略纳什均衡，下图（图 4.8）为 Nashsweeper 服务端工作以及 Web API：

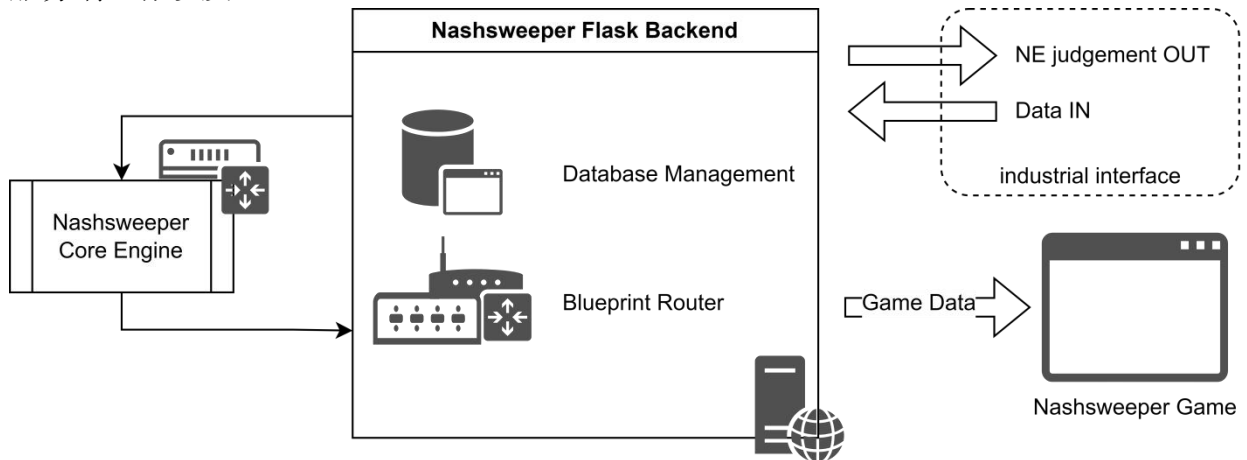


图 4.8: Nashsweeper Back-end

4.3.3 Nashsweeper 基于 Axios^[35] 前后端数据交互

由于 Nashsweeper 采用客户端与服务端 Flask^[36] 框架分离式开发，因此前后端通信采用了对 AJAX 进行异步封装的 Axios，下图（图 4.9）为前后端数据传输：

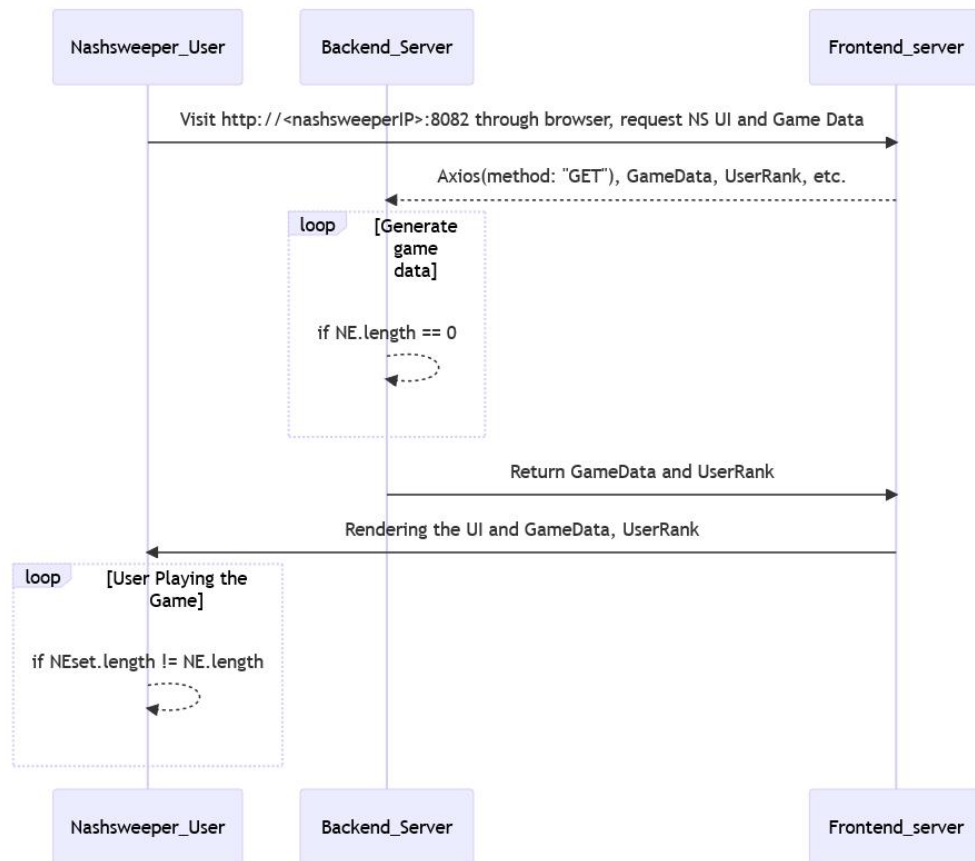


图 4.9: Nashsweeper 前后端数据交互

4.3.4 Nashsweeper 基于 Pinia^[37] 的前端组件间通信

Nashsweeper 游戏在进行的工程中，用户的 checkerboard 点击事件与下方的状态显示与游戏日志等数据需要进行状态同步，由于很多数据隶属于不同组件，若采用 Vue 中的子组件向父组件提交数据，然后父组件再分发给其他需要该数据的子组件的话，数据同步效率过低，因此 Nashsweeper 采用了 Pinia 对组件间的数据进行统一存储。

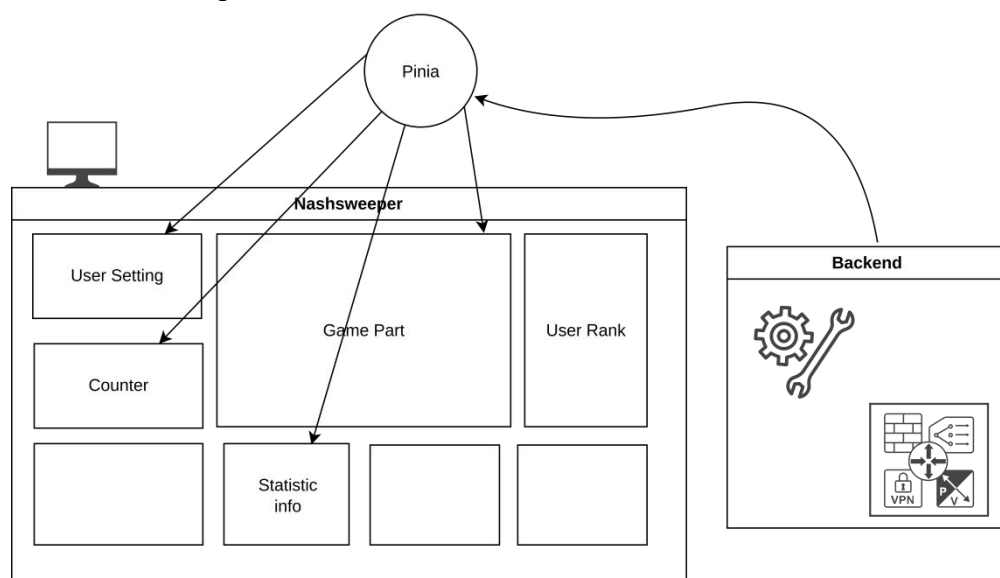









图 4.10: Nashsweeper 组件间通信

4.4 棋盘中每个单元格状态

由于 Nashsweeper 是一个单机游戏,用户在使用的时候只需要通过点击棋盘就可以了,游戏的主要目标是找到最佳响应 (Best Response)、纳什均衡 (Nash Equilibrium)。因此在本文中我们设定棋盘中的单元格为以下四种状态 (表 4.1) :

表 4.1: 单元格的四种状态

单元格显示状态 (Displace of Cells)	描述 (Description)
	(Unrevealed) 单元格尚未被点击
	(Revealed) 单元格已被点击
	(BR) 单元格已被点击, 且为最佳响应
	(NE) 单元格已被点击, 且为纳什均衡

当用户点击一个 Unrevealed 状态单元格的时候, Nashsweeper 游戏需要去标记该单元格为 revealed、BR、NE 三种状态。让 (s_1, s_2, \dots, s_n) 代表当前 Unrevealed 单元格的策略, 如果经过计算, 其中的所有 Regret function (r_1, r_2, \dots, r_n) 都为 0, 那么该单元格为纳什均衡, 被标记为 ; 如果 (r_1, r_2, \dots, r_n) 其中存在 $r_i=0, (i \in n)$, 则该单元格为最佳响应, 颜色标记为 , 如果一个单元格既不属于 NE 也不属于 BR, 那么该单元格仅为 Revealed, 被标记为 。实现上述将 unrevealed 单元格转化为 Revealed、BR、NE 状态的算法伪代码如 Algorithm 2 所示:

Algorithm 2: Sweep operation process.

Input: $(s_1, s_2, \dots, s_n), (u_1, u_2, \dots, u_n), (r_1, r_2, \dots, r_n)$

1. if $r_i == 0$ for all i then // (s_1, s_2, \dots, s_n) is a Nash equilibrium
 2. mark $u_i(s_i, s_{-i})$ as a Nash equilibrium;
 3. for $i \in n$ do
 4. for $i' \in n$ and $i' \neq n$ do
 5. mark $u_i(s_i, s_{-i})$ as a revealed cell; // sweep operation
 6. end
 7. end
 8. else if there exists $r_i == 0$ then // (s_1, s_2, \dots, s_n) is a best response
 9. for $i \in n$ do
 10. if $r_i == 0$ then
 11. mark $u_i(s_i, s_{-i})$ as a best response;
 12. for $i \in n$ and $i' \neq n$ do
 13. mark $u_i(s_i, s_{-i})$ as an revealed cell; // sweep operation
 14. end
 15. end
 16. end
-

```

17. else
18.   mark  $u_i(s_i, s_{-i})$  as a revealed cell;
19. end
    
```

4.5 游戏通关条件

在 Nashsweeper 游戏中，有三种级别的难度可供选择：入门级别、中级、专家级别，它们对应这不同难度的通关条件。下面是三个通关条件的描述：

- 面向入门级玩家：如果用户可以找到一个纳什均衡，则游戏结束。
- 面向中级玩家：如果用户能找到全部纳什均衡，则游戏结束。
- 面向专家级玩家：如果用户能找到全部纳什均衡与最佳响应，则游戏结束。

4.6 游戏流程图

本小节描述 Nashsweeper 游戏的概述。在 Nashsweeper 游戏的开始阶段，用户需要确定 Nashsweeper 游戏的难度等级。然后，用户可以选择生成新游戏或加载历史数据。接下来，用户需要选择一个单元格并单击它。单击单元格后，将进行扫描操作过程。

Nashsweeper 游戏将一直持续到满足游戏结束条件为止。下图(图 4.11)显示了 Nashsweeper 游戏的流程图。

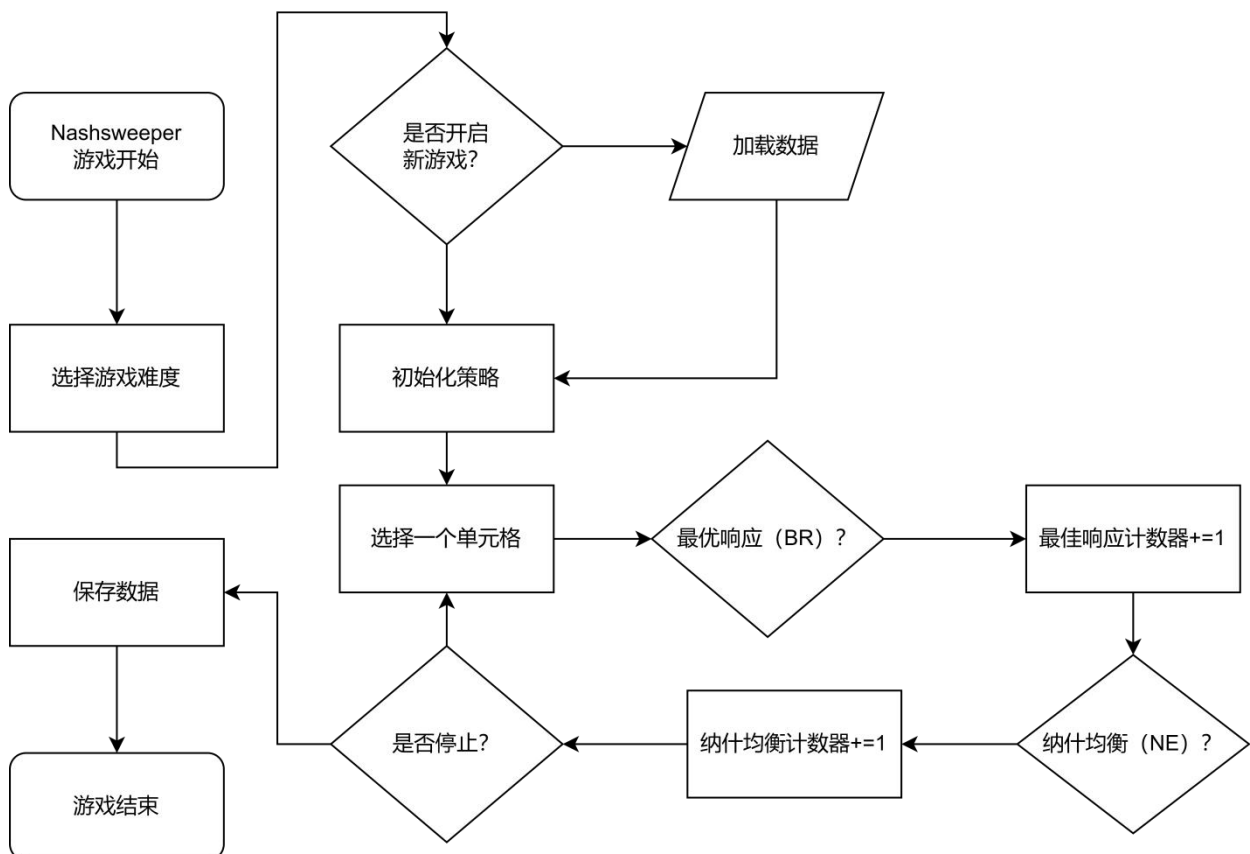


图 4.11：游戏运行流程图

5 案例分析

在本部分，我们将介绍一个 Nashsweeper 游戏的案例研究来说明 Nashsweeper 游戏的游戏过程。

5.1 输入数据

在这个案例研究(表 5.1)中，有两个玩家，每个玩家有 8 个策略。收益值(Payoff value)是在 0 和 100 的范围内随机产生的。每个玩家的目标都是最大化其回报价值。下表显示了 SQLite 数据库中两个玩家非合作游戏的示例。通过使用 Algorithm 1，我们得到了在(图 5.1，图 5.2)中所示的 regret value。

表 5.1: 在 SQLite 数据库中的两个玩家非合作游戏的一个案例

ID	Strategy		Payoff Value		ID	Strategy		Payoff Value	
	玩家 1	玩家 2	玩家 1	玩家 2		玩家 1	玩家 2	玩家 1	玩家 2
1_1	1	1	76	97	5_1	5	1	50	68
1_2	1	2	72	67	5_2	5	2	69	52
1_3	1	3	25	84	5_3	5	3	22	59
1_4	1	4	73	46	5_4	5	4	57	37
1_5	1	5	93	20	5_5	5	5	60	5
1_6	1	6	12	44	5_6	5	6	1	93
1_7	1	7	68	93	5_7	5	7	55	64
1_8	1	8	20	87	5_8	5	8	45	51
2_1	2	1	43	24	6_1	6	1	46	18
2_2	2	2	68	44	6_2	6	2	87	12
2_3	2	3	62	97	6_3	6	3	32	0
2_4	2	4	72	66	6_4	6	4	37	85
2_5	2	5	96	16	6_5	6	5	43	3
2_6	2	6	42	3	6_6	6	6	70	93
2_7	2	7	14	44	6_7	6	7	53	62
2_8	2	8	51	37	6_8	6	8	26	18
3_1	3	1	63	40	7_1	7	1	80	8
3_2	3	2	76	86	7_2	7	2	85	13
3_3	3	3	43	66	7_3	7	3	54	80
3_4	3	4	13	63	7_4	7	4	37	43
3_5	3	5	6	92	7_5	7	5	23	11
3_6	3	6	15	15	7_6	7	6	91	48
3_7	3	7	68	19	7_7	7	7	88	57
3_8	4	8	69	77	7_8	7	8	53	25
4_1	4	1	49	90	8_1	8	1	71	90
4_2	4	2	29	43	8_2	8	2	35	57
4_3	4	3	75	93	8_3	8	3	87	98
4_4	4	4	18	70	8_4	8	4	57	32
4_5	4	5	0	31	8_5	8	5	41	25
4_6	4	6	99	96	8_6	8	6	28	94
4_7	4	7	5	71	8_7	8	7	100	14
4_8	4	8	4	5	8_8	8	8	68	53

5.2 Regret Value 计算

利用下图（图 5.1，图 5.2）所示的结果，我们可以找到所有的最佳响应和纳什均衡，如下表所示。在下表（表 5.2）中，具有灰色背景的行代表纳什均衡。

4	15	62	0	3	87	32	49
37	19	25	1	0	57	86	18
17	11	44	60	90	84	32	0
31	58	12	55	96	0	95	65
30	18	65	16	46	98	45	24
34	0	55	36	53	29	47	43
0	2	33	36	73	8	12	16
9	52	0	16	55	71	0	1

图 5.1: 玩家 1 所有策略的 Regret Value

0	30	13	51	77	53	4	10
73	53	0	31	81	94	53	60
52	6	26	29	0	77	73	15
6	53	3	26	65	0	25	91
25	41	34	56	88	0	29	42
75	81	93	8	90	0	31	75
72	67	0	37	69	32	23	55
8	41	0	66	73	4	84	45

图 5.2: 玩家 2 所有策略的 Regret Value

表 5.2: Best Response 与纳什均衡

ID	策略		收益值		Regret 值		结果
	玩家 1	玩家 2	玩家 1	玩家 2	玩家 1	玩家 2	
1_1	1	1	76	97	4	0	BR
1_4	1	4	73	46	0	51	BR
2_3	2	3	62	97	25	0	BR
2_5	2	5	96	16	0	81	BR
3_6	3	6	6	92	90	0	BR
3_8	3	8	69	77	0	15	BR
4_6	4	6	99	96	0	0	NE
5_6	5	6	1	93	98	0	BR
6_2	6	2	87	12	0	81	BR
6_6	6	6	70	93	29	0	BR
7_1	7	1	80	8	0	72	BR
7_3	7	3	54	80	33	0	BR
8_3	8	3	87	98	0	0	NE
8_7	8	7	100	14	0	84	BR

5.3 初始化用户图形界面

在加载数据后，下图（图 5.1）给出了一个 Nashsweeper 游戏的图形用户界面示例。

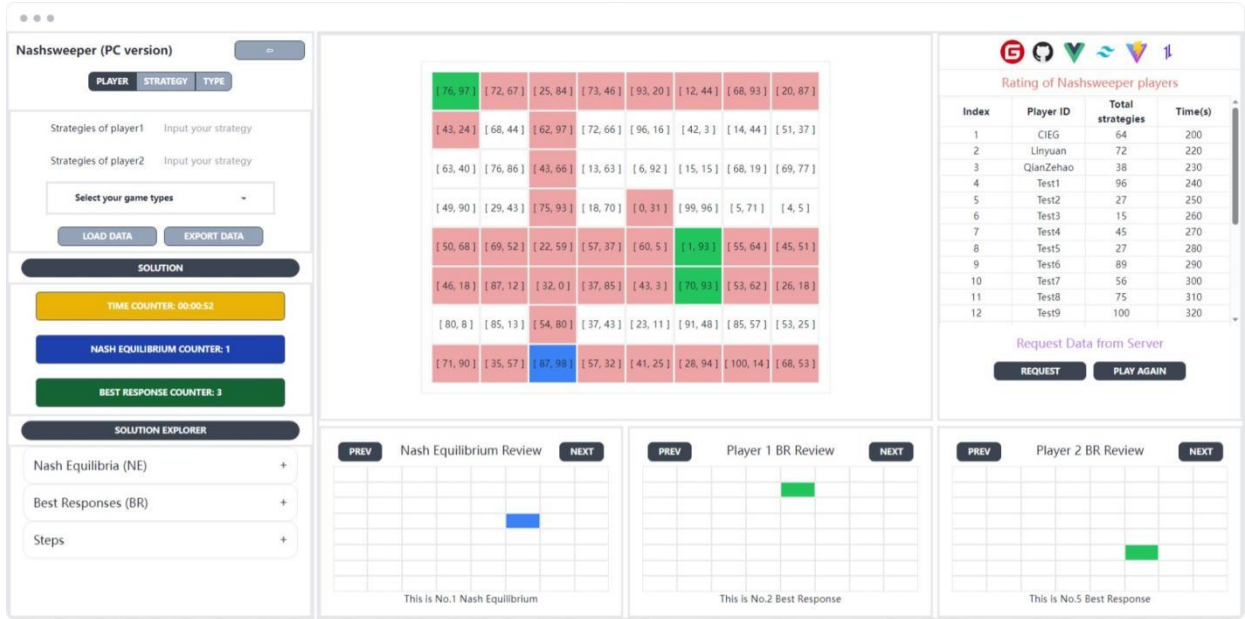


图 5.1: Nashsweeper 游戏截图

5.4 开始游戏

在本小节中，我们将使用表中显示的数据作为输入数据，介绍 Nashsweeper 游戏玩过程。在加载输入数据后，我们可以看到下图（图 5.2）中的 Nashsweeper 游戏的初始状态。

假设用户通过依次点击图 6 中所示的单元格来玩 Nashsweeper 游戏。[72, 67]不是一个最好的反应或纳什均衡。因此[72, 67]是一个 Revealed 的单元格。[76, 97]是玩家 2 的最佳响应，第一行除了[76, 97]外都被标记为 Revealed 的单元格。[69, 77]是最佳响应，玩家 1 和最后一列的响应被标记为已显示的单元格。[87, 98]是一种纳什均衡。第三列和最后一行被标记为 Revealed 的单元格。[85, 13]不是一个最佳响应或纳什均衡。只有[85, 13]是一个 Revealed 的单元格。[87, 12]是玩家 1 的最佳响应，第二列被标记为已显示的单元格。[80, 8]是玩家 1 的最佳响应，因此第一列其他单元格被标记为 Revealed 单元格。

(72, 67) → (76, 97) → (69, 77) → (87, 98)



(99, 96) ← ... ← (80, 8) ← (87, 12) ← (85, 13)

图 5.2: Nashsweeper 游戏的单元格点击序列

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]

点击 [72, 67]

[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

点击 [76, 97]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

点击 [69, 77]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

点击 [87, 98]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]

点击 [53, 13]

[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

点击 [87, 12]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

点击 [80, 8]

[76,97]	[72,67]	[25,84]	[73,46]	[93,20]	[12,44]	[68,93]	[20,87]
[43,24]	[68,44]	[62,97]	[72,66]	[96,16]	[42,3]	[14,44]	[51,37]
[63,40]	[76,86]	[43,66]	[13,63]	[6,92]	[15,15]	[68,19]	[69,77]
[49,90]	[29,43]	[75,93]	[18,70]	[0,31]	[99,96]	[5,71]	[4,5]
[50,68]	[69,52]	[22,59]	[57,37]	[60,5]	[1,93]	[55,64]	[45,51]
[46,18]	[87,12]	[32,0]	[37,85]	[43,3]	[70,93]	[53,62]	[26,18]
[80,8]	[85,13]	[54,80]	[37,43]	[23,11]	[91,48]	[88,57]	[53,25]
[71,90]	[35,57]	[87,98]	[57,32]	[41,25]	[28,94]	[100,14]	[68,53]

最终状态

6 结论及未来发展方向

6.1 Nashsweeper 的应用前景

在教学上, Nashsweeper 游戏如今已经被郑州大学管理学院开设的课程采用, 如近期管理学院负责的数学建模培训, 同学们在学习博弈论相关课程时, 非常高兴和积极地参与使用 Nashsweeper。Nashsweeper 可以帮助用户简单快速理解纳什均衡的计算过程, 非合作博弈有囚徒困境、性别之战、零和博弈等非合作博弈类型, 这些问题都可以通过 Nashsweeper 解决, 帮助用户学习不同类型的非合作博弈。

在工业上, Nashsweeper 的计算核心 Nashsweeper-core-engine 作为 Nashsweeper 的 CLI 版本以及 Nashsweeper Web API 可用于计算超大规模纯策略纳什均衡。这将能解决很多工业上的问题。在工业企业, 若面对最优控制、企业博弈、库存管理等问题时(例如, 在一个简单的情景中, 假设制造商和零售商都有两种定价策略: 高价和低价。如果制造商和零售商都选择高价, 则他们都会获得最小利润; 如果他们都选择低价, 则他们都会获得最大利润; 如果制造商选择高价而零售商选择低价, 则制造商会获得最小利润而零售商会获得最大利润; 如果制造商选择低价而零售商选择高价, 则制造商会获得最大利润而零售商会获得最小利润。因此, 在这种情况下, 纳什均衡是制造商和零售商都选择低价。)类似问题可以建模为博弈论, 通过调用 Nashsweeper 接口去解决该类问题。

6.2 Nashsweeper 的限制

Regret-base 算法的劣势: regret-base 算法是暴力枚举法, 时间复杂度为 $O(n^2)$, 在进行纯策略纳什均衡精确解搜索时, 消耗时间长。Nashsweeper 博弈的另一个局限性是不支持数学规划方法来获得纯纳什均衡。

Nashsweeper 仅支持单机模式: 通常, 博弈论的应用至少需要两个参与者来玩游戏。在本文中, Nashsweeper 是一款单机游戏, 只是不支持多人同时玩。Nashsweeper 游戏只关注纯纳什均衡的计算, 限制了两个或更多玩家之间交互的功能。Nashsweeper 游戏的用户在玩 Nashsweeper 游戏并知道所有玩家的 Payoff 时, 他/她处于上帝视角。因此, Nashsweeper 游戏的用户不会体验到玩家之间的交互。

6.3 Nashsweeper 的未来发展方向

进一步关于 Nashsweeper 的研究可以扩展到以下几个方面: (1) 多人同时玩游戏; (2) 本文设计了一款基于 PC 的 Nashsweeper 游戏。未来可能会开发其他基于平台的 Nashsweeper 游戏, 例如 iOS 或 Android。

支持多人模式: 针对单机游戏的优化, 通过引入预训练的机器人, 模拟对手, 与用户进行游戏, 提升 Nashsweeper 的难度, 增强用户体验。

更多平台支持: Nashsweeper 未来可能会开发其他基于平台的 Nashsweeper 游戏, 例如 iOS 或 Android, 同时针对移动端的小屏幕 UI 进行优化: 减少棋盘单元格个数等。

对于 Regret 算法的优化加速: 在本文中, Regret 算法在计算 $\max_{s_i \in S_i} (s'_i, s_{-i})$ 时, 虽然使

用到了 NumPy 进行矩阵运算优化, 但运行时依旧是单线程, 未来将基于硬件加速平台如 GPU、FPGA 等开发异构加速。

致 谢

行文至此，意味着我的本科生涯已至谢幕时刻。回首四年间郑州大学生活，思绪泉涌，在这里写下论文最后的致谢。我深知我在这里的每一点成绩，每一次获奖，以及每一个高光时刻，都离不开我身边的人，你们的善意的关心、一贯的支持以及无私的帮助造就了如今的我，在此，我要向你们表示最衷心的感谢！

首先，我要感谢我的导师周艳杰，他不仅仅是一位卓越的启蒙者，还是我在学术和计算机技术方面的引路人，感谢老师的器重，让我可以加入计算机与工业工程小组，与 CIEG 很多优秀的成员共同进步，在两年多时间里，他的项目让我大学变的充实，也让我在工业工程科研能力方面得到了全面而深入的提高。他严谨求实的学风、博学多才的学识、开阔创新的思维以及平易近人的处世原则都深深地影响了我，并将成为我今后工作和生活中不断追求和学习的目标。与此同时，感谢赵倩倩老师对我的毕业论文提供的宝贵意见以及指导，在思路拓展方面给予了我很多启发；感谢李尽法、叶正梗老师对于我本科期间创新创业比赛的指导；感谢工业工程所有老师在大学四年间对我的教诲，通过教授管理工程学院最硬核的知识，实现了我一直以来的技术梦。

其次，我要感谢我的父母和家人，在我的求学生涯中，他们始终是最坚强的后盾和最温暖的港湾，时刻关注我的身心健康。作为家中的独子，我和父母每天都会通话交流，在专业选择、出国深造、未来职业规划方面他们支持着我的每一个决定，从不干预我的选择，聆听我每一个天马行空的想法，同时，他们在我迷茫困惑时也给予了我指引和建议，在我失意时鼓励我，为我的成长创造了良好的物质和精神条件，让我可以专注于自己的目标，不需要为其他事物分心。

其次，我要感谢我的同学和朋友们（同窗室友马少杰、王晓、黄川泽；青年学术创新中心 17-20 级的朋友：付婧、未思艺、于小淞等；OpenIE 团队：张紫阳、罗雯杰；相师的好兄弟于弋航、詹子煜；在 CIEG 时认识的师兄师姐们；以及其他在大学参加比赛/工作时一起共事的朋友们），他们在四年间也深深影响了我，见证过我的辉煌，是我最珍惜的朋友，也是我最宝贵的财富。在此向他们表示最诚挚的祝福，祝前程似锦！

郑州大学以及管理工程学院培养了我对于科学真理的敬畏之心，以及对于先进技术的追求，这让我有了勇攀高峰的决心，驱动着我即将远赴英国继续深造，开启不屈远征。立于皓月之边，不弱星光之势。

附录 A: Nashsweeper 核心计算引擎

```

1. # =====
2. # Nashsweeper Core Computing Engine
3. # Using numpy, sqlite3 and itertools
4. # =====
5. from itertools import product
6. # import numpy
7. import sqlite3
8. # =====
9.
10. # this part is used to generate id lst.
11. # User X has n strategies, and his strategy list index is from 0 to n-1.
12.
13.
14. def atom_lst_gen(n):
15.     lst = [ for in range(n)]
16.     return lst
17.
18.
19. def users_strategy_lst_gen(UserStrategyNum):
20.     users_strategy_lst = []
21.     for in UserStrategyNum:
22.         users_strategy_lst.append(atom_lst_gen(_))
23.     return users_strategy_lst
24.
25. # generate strategy_combination by using iterator
26.
27.
28. def strategy_combination_lst_gen(users_strategy_lst):
29.     strategy_combination = product(*users_strategy_lst)
30.     strategy_combination = list(strategy_combination)
31.     return strategy_combination
32.
33.
34. def id_lst_gen(strategy_combination):
35.     id_lst = []
36.     for strategy in strategy_combination:
37.         strategy = " ".join(map(str, strategy))
38.         id_lst.append(strategy)
39.     return id_lst
40. # =====
41.
42.
43. # =====
44. # this part is related to database
45. # If here exists N gamers, it will generate a column for each user to store their payoff.
46. def valst_gen(N):
47.     valst_pre = []
48.     for _ in range(N):
49.         str_atom = 'Payoff' + str( ) + ' + ' + 'NUMBER'
50.         valst_pre.append(str_atom)
51.     valst = ". ".join(valst_pre)
52.     return valst
53.
54.
55. class SQLiteProcess(object):
56.     def __init__(self, N, Dataset):

```

```

57.     self.N = N
58.     self.Dataset = Dataset
59.     # create the database, id, Payoff1, Payoff2,..., PayoffN
60.
61.     def create_database(DBname, TBname, N):
62.         DBname += '.db'
63.         valst = valst_gen(N)
64.         id_info = 'id TEXT PRIMARY KEY,'
65.         conn = sqlite3.connect(DBname)
66.         cursor = conn.cursor()
67.         executeStr = 'CREATE TABLE IF NOT EXISTS ' + \
68.             TBname + '(' + id_info + valst + ')'
69.         cursor.execute(executeStr)
70.         conn.commit()
71.         conn.close()
72.     # insert data(id, Payoff1, Payoff2,..., PayoffN) into database
73.
74.     def insert_data(DBname, TBname, N, Dataset):
75.         valst_pre = []
76.         for in range(N):
77.             str_atom = 'Payoff' + str( ) + ''
78.             valst_pre.append(str_atom)
79.         valst = ','.join(valst_pre)
80.         insertTab = 'id, ' + valst
81.         DBname += '.db'
82.         val_pre = tuple(['?' for i in range(N+1)])
83.         val = ','.join(val_pre)
84.         conn = sqlite3.connect(DBname)
85.         cursor = conn.cursor()
86.         executeStr = 'INSERT INTO' + ' ' + TBname + \
87.             ' ' + '(' + insertTab + ')' + ' ' + ' ' + \
88.             'VALUES' + ' ' + '(' + val + ')'
89.         cursor.executemany(executeStr, Dataset)
90.         conn.commit()
91.         conn.close()
92.
93.     def search_data(DBname, TBname, keyword, id):
94.         DBname += '.db'
95.         conn = sqlite3.connect(DBname)
96.         cursor = conn.cursor()
97.         executeStr = 'SELECT' + ' ' + keyword + ' ' + 'FROM' + ' ' + TBname + \
98.             ' ' + 'WHERE' + ' ' + 'id' + '=' + '?'
99.         cursor.execute(executeStr, (id,))
100.         result = cursor.fetchall()
101.         conn.close()
102.         return result[0][0]
103. # =====
104.
105.
106. # =====
107. # Regret Value Calculation
108. # =====
109.
110. if __name__ == "__main__":
111.     Dataset = [
112.         ('0_0_0', 1, 2, 3),
113.         ('0_0_1', 4, 5, 6),
114.         ('0_1_0', 12, 42, 34),
115.         ('0_1_1', 34, 22, 27),

```

```

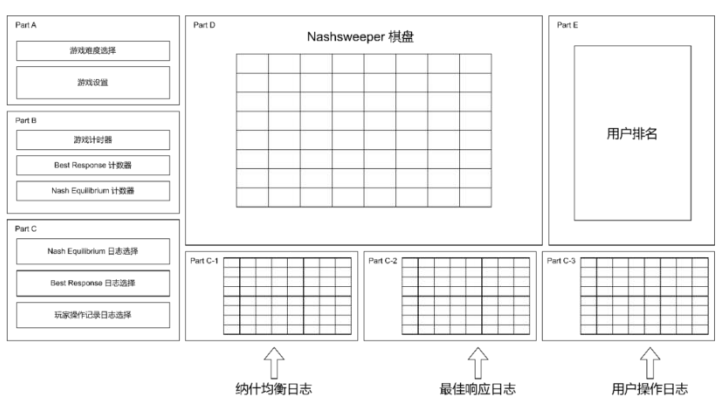
116.     ('1_0_0', 42, 56, 23),
117.     ('1_0_1', 93, 26, 6),
118.     ('1_1_0', 23, 68, 54),
119.     ('1_1_1', 57, 23, 11)
120. ]
121. # SQLiteProcess.create_database('TestDB', 'TestTB', 3)
122. # SQLiteProcess.insert_data('TestDB', 'TestTB', 3, Dataset)
123. print(SQLiteProcess.search_data('TestDB', 'TestTB', 'Payoff1', '0_0_0'))
124. # print(SQLiteProcess.search_data('TestDB', 'TestTB', '0_1_0'))
125.
126. # test_strategy_matrix = [[76, 97], [72, 67], [25, 84],
127. # [73, 46], [93, 20], [12, 44], [68, 93], [20, 87], [43, 24],
128. # [68, 44], [62, 97], [72, 66], [96, 16], [42, 3], [14, 44],
129. # [51, 37], [63, 40], [76, 86], [43, 66], [13, 63],
130. # [6, 92], [15, 15], [68, 19], [69, 77], [49, 90], [29, 43],
131. # [75, 93], [18, 70], [0, 31], [99, 96], [5, 71], [4, 5],
132. # [50, 68], [69, 52], [22, 59], [57, 37], [60, 5], [1, 93],
133. # [55, 64], [45, 51], [46, 18], [87, 12], [32, 0], [37, 85],
134. # [43, 3], [70, 93], [53, 62], [26, 18], [80, 8], [85, 13],
135. # [54, 80], [37, 43], [23, 11], [91, 48], [85, 57], [53, 25],
136. # [71, 90], [35, 57], [87, 98], [57, 32], [41, 25], [28, 94], [100, 14], [68, 53]]
137.
138. # test_strategy_matrix_array = numpy.array(
139. #     test_strategy_matrix).reshape(8, 8, 2)
140.
141. # data_input_size = list(test_strategy_matrix_array.shape)
142. # UserStrategyNum = data_input_size[-1]
143. # N = data_input_size[-1]
144.
145. # users_strategy_lst = users_strategy_lst_gen(UserStrategyNum)
146. # strategy_combination = strategy_combination_lst_gen(users_strategy_lst)
147. # id_lst = id_lst_gen(strategy_combination)
148. ## print(id_lst)
149.
150. # DatabaseData = []
151. # for in range(len(id_lst)):
152. #     tupChild = (id_lst[_], test_strategy_matrix[_
153. #         [0], test_strategy_matrix[_][1])
154. #     DatabaseData.append(tupChild)
155.
156. # print(DatabaseData)

```

附录 B: Nashsweeper 使用文档


Nashsweeper User Manual

1. Nashsweeper 界面认识



Part A: 游戏难度选择, 游戏设置
Part B: 游戏计时器, Best Response 计数器, Nash Equilibrium 计数器
Part C: Nash Equilibrium 日志选择, Best Response 日志选择, 玩家操作日志选择
Part D: Nashsweeper 棋盘
Part E: 用户排名
Part C-1: 纳什均衡日志
Part C-2: 最佳响应日志
Part C-3: 用户操作日志

2. 用户数据上传



http://<ns_server_IP>:5000/UploadPage/uploadfile
支持Database/CSV/List格式输入

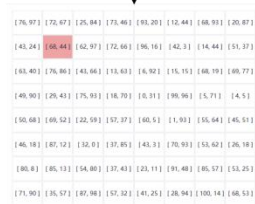
3. 游戏规则说明

Nashsweeper的目的只有一个，用最快速度找到棋盘中的所有纳什均衡点！

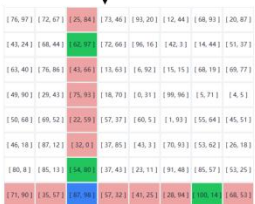
棋盘中的单元格有如下染色方式：

单元格显示状态 (Displace of Cells)	描述 (Description)
	(Unrevealed) 单元格尚未被点击
	(Revealed) 单元格已被点击
	(BR) 单元格已被点击，且为最佳响应
	(NE) 单元格已被点击，且为纳什均衡


当点击一个Unrevealed单元格时



当点击一个NE单元格时

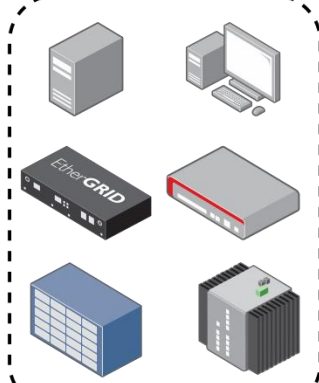


当点击一个BR单元格时(76, 97)



NS的跨平台部署

Nashsweeper支持多平台部署，我的目标是有芯片的地方就可以运行Nashsweeper（目前支持x86架构服务器、个人计算机、M2芯片MacBook，如果你足够能折腾的话，也可以在如下设备上部署它：家用NAS、STM32芯片的机顶盒，Raspberry Pi.....）



可以运行Nashsweeper的生态

Step 1: Git NS官方仓库

git clone https://gitee.com/qian_zehao/nashsweeper.git

Step 2: 安装Docker和docker-compose


Step 3: sudo docker-compose build

Step 4: sudo docker-compose up -d

面向工业和数据科学

Nashsweeper用于计算博弈双方NE和BR的核心计算引擎，亦可用于数据科学领域，以及工业场景中，处理超大规模纯策略纳什均衡问题，扫描下方二维码，为Nashsweeper core在Jupyter Notebook中的调用

Nashsweeper for Industry



参考文献

- [1] Nash J. Non-cooperative games[J]. Annals of mathematics, 1951:286–295.
- [2] Shubik M. On gaming and game theory[J]. Management Science, 1972, 18(5-part-2):37-53.
- [3] Davis M, Maschler M. The kernel of a cooperative game[J]. Naval Research Logistics Quarterly, 1965, 12(3):223–259.
- [4] 许吉祥,侯剑,谭彦华等.求解广义纳什均衡问题的指数型惩罚函数方法[J].运筹与管理,2015,24(01):81-88.
- [5] Bruno Fanzeres, A Column-and-Constraint Generation Algorithm to Find Nash Equilibrium in Pool-Based Electricity Markets[J]. Electric Power System Research, 2020, 12:106806.
- [6] 陈盼华. 广义纳什均衡的一类优化方法[D].郑州大学,2015.
- [7] 王琳蒙,王玉惠,陈谋,刘昊天.基于改进麻雀算法的非完备信息博弈策略研究[J].吉林大学学报(信息科学版),2022,40(4):589-599.
- [8] 裴小兵,王诗慧.基于博弈人工蜂群算法的多目标车间调度研究[D].武汉大学学报(工学版),2023.
- [9] 张国强,赵国党.一种改进的微分进化算法求解纳什均衡问题与广义纳什均衡问题[J].运筹与管理,2023.
- [10] Zhou Y, Kim KH. A game theoretic model and a coevolutionary solution procedure to determine the terminal handling charges for container terminals[J]. Computers & Industrial Engineering, 2022, 144:106466.
- [11] 赵倩. 基于 Q-learning 的多智能体系统最优一致性的研究[D].天津大学,2019.
- [12] 魏娜,刘明雍.基于贝叶斯纳什均衡的不完全信息博弈目标分配决策[J].西北工业大学学报,2022,4:40.
- [13] 侯剑.基于近似重构的广义 Nash 均衡问题的数值方法[D].大连理工大学,2013.
- [14] 李慧敏.基于群智能算法的 Nash 平衡的计算与实现[D].贵州大学,2022.
- [15] 李治辰.通信受限下的多智能体系统协同控制研究[D].燕山大学,2021.
- [16] 李佳莲.样本高效的强化学习方法研究[D].清华大学,2021.
- [17] Anema J. Game theory[EB/OL].(2021-08-25), [2023-04-21], <http://pi.math.cornell.edu/~mec/2008-2009/Anema/games.html>.
- [18] Fan CP. Teaching children cooperation -- an application of experimental game theory[J]. Journal of Economic Behavior & Organization 2000, 41(3):191–209.
- [19] Rothman SB. Developing and adapting simulations through six points of variance: An example of teaching applied game theory through international negotiations[J]. International Studies Perspectives, 2012, 13(4):437–457.
- [20] Ghic G, Grigorescu CJ. Applications of games theory in analyzing teaching process[J]. Procedia - Social and Behavioral Sciences, 2014, 116:3588–3592.
- [21] Prisner E. Game theory through examples[J], American Mathematical Soc, 2014:volume 46.
- [22] Badeau S, Deloche R, Koscielniak T. Online game-enhanced teaching in game theory[J]. EUNIS Journal of Higher Education, 2014:1–10.
- [23] O’Roark B, Grant W. Games superheroes play: Teaching game theory with comic book favorites[J]. The Journal of Economic Education, 2018, 49(2):180–193.
- [24] Abasian F, Ronnqvist M, Marier P, Fjeld D. Game--the transportation game[J]. INFORMS Transactions on Education, 2020, 21(1):1–12.

- [25]Iranpoor M. Knights exchange puzzle-teaching the efficiency of modeling[J]. INFORMS Transactions on Education. 2021, 21(2):108–114.
- [26]Wu Z, Dang C, Karimi HR, Zhu C, Gao Q. A mixed 0-1 linear programming approach to the computation of all pure-strategy nash equilibria of a finite n-person game in normal form. Mathematical Problems in Engineering 2014, 5:1-8.
- [27]Corley H, et al. A regret-based algorithm for computing all pure nash equilibria for noncooperative games in normal form[J]. Theoretical Economics Letters, 2020, 10(06):1253.
- [28]McKelvey RD, McLennan AM, Turocy TL Gambit: Software tools for game theory[EB/OL].(2006), [2023-3-22], <http://www.gambit-project.org/>
- [29]Vue 3.0 Official Doc[EB/OL]. (2023-04-16), [2023-05-02], <https://vuejs.org/>
- [30]NumFOCUS, Inc. Pandas[EB/OL].(2023-02-24), [2023-04-18], <https://pandas.pydata.org/>
- [31]NumPy Team. NumPy[EB/OL].(2023-01-20), [2023-03-22], <https://numpy.org/>
- [32]Docker, Inc. Docker Document[EB/OL]. (2023-01-25), [2023-04-18], <https://www.docker.com/>
- [33]Kaye R. Minesweeper is np-complete[J]. The Mathematical Intelligencer 2000, 22(2):9-15.
- [34]Mahato S, Yadav DK, Khan DA. A minesweeper game-based steganography scheme[J]. Journal of Information Security and Applications, 2017, 32:1-14.
- [35]Matt Zabriskie. Axios -- Promise based HTTP client for the browser and node.js[EB/OL]. (2023-03-05), [2023-4-10], <https://axios-http.com/>
- [36]Python Flask Official Site[EB/OL]. [2023-3-28], <https://flask.palletsprojects.com/en/2.3.x/>
- [37]Pinia -- The intuitive store for Vue.js, [EB/OL]. (2023-03-17), [2023-04-10], <https://pinia.vuejs.org/zh/>