МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

УРБАНОВИЧ

Дмитрий Игоревич

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ И ЗАДАЧА ОПТИМИЗАЦИИ ИГРЫ КАРКАССОН

Дипломная работа

Научный руководитель:
ст. преподаватель И. И. Козлов

Допущен к защите
«___» _____ 2025 г.
Зав. кафедрой дифференциальных уравнений и системного анализа
канд. физ.-мат. наук, доцент Л. Л. Голубева

Минск, 2025

ОГЛАВЛЕНИЕ

РЕФЕРАТ

В дипломной работе __ страниц, __ иллюстраций, __ источников, __ приложения.

СТОХАСТИЧЕСКАЯ ИГРА, ОПТИМИЗАЦИЯ СТРАТЕГИИ, РАЗМЕЩЕНИЕ ТАЙЛОВ, LLM, MCP, PYTHON.

Объектом исследования дипломной работы является стохастическая игра с полной информацией на примере игры Каркассон.

Целью дипломной работы является разработка гибридного алгоритма оптимизации стратегии, объединяющего символьные вычисления (MCP) и эвристический поиск на базе LLM.

Для достижения поставленной цели были использованы: язык программирования Python, Model Context Protocol (MCP), а также большая языковая модель (LLM) через Ollama.

В дипломной работе получены следующие результаты:

1. Разработан игровой движок и MCP-сервер для вычисления допустимых ходов.
2. Реализована интеграция с LLM для генерации стратегических гипотез и выбора приоритетов.
3. Проведен анализ применимости LLM в качестве эвристической функции для задач комбинаторной оптимизации на графах.

Дипломная работа является завершенной, поставленные задачи решены, присутствует возможность дальнейшего развития подхода на другие настольные игры.

Дипломная работа выполнена автором самостоятельно.

РЭФЕРАТ

У дыпломнай працы __ старонак, __ малюнкаў, __ крыніц, __ дадаткаў.

СТАХАСТЫЧНАЯ ГУЛЬНЯ, АПТЫМІЗАЦЫЯ СТРАТЭГІІ, РАЗМЯШЧЭННЕ ТАЙЛАЎ, LLM, MCP, PYTHON.

Аб'ектам даследвання дыпломнай працы з'яўляецца стахастычная гульня з поўнай інфармацыяй на прыкладзе гульні Каркасон.

Мэтай дыпломнай працы з'яўляецца распрацоўка гібрыднага алгарытму аптымізацыі стратэгіі, які аб'ядноўвае сімвальныя вылічэнні (MCP) і эўрыстычны пошук на базе LLM.

Для дасягнення пастаўленай мэты выкарыстоўваліся: мова праграмавання Python, Model Context Protocol (MCP), а таксама вялікая моўная мадэль (LLM) праз Ollama.

У дыпломнай працы атрыманы наступныя вынікі:

1. Распрацаваны гульнявы рухавічок і MCP-сервер для вылічэння дапушчальных хадоў.
2. Рэалізавана інтэграцыя з LLM для генерацыі стратэгічных гіпотэз і выбару прыярытэтаў.
3. Праведзены аналіз прымяняльнасці LLM у якасці эўрыстычнай функцыі для задач камбінаторнай аптымізацыі на графах.

Дыпломная праца з'яўляецца завершанай, пастаўленыя задачы вырашаны, прысутнічае магчымасць далейшага развіцця падыходу на іншыя настольныя гульні.

Дыпломная праца выканана аўтарам самастойна.

ABSTRACT

Thesis project is presented in the form of an explanatory note of __ pages, __ figures, __ references, __ applications.

STOCHASTIC GAME, STRATEGY OPTIMIZATION, TILE PLACEMENT, LLM, MCP, PYTHON.

The research object of this thesis project is a stochastic game with full information, using the game Carcassonne as an example.

The purpose of the thesis is to develop a hybrid strategy optimization algorithm combining symbolic computation (MCP) and LLM-based heuristic search.

To achieve the goal, the following were used: Python programming language, Model Context Protocol (MCP), as well as a large language model (LLM) via Ollama.

The main results of the thesis project are as follows:

1. A game engine and MCP server were developed to compute valid moves.
2. Integration with LLM was implemented to generate strategic hypotheses and select priorities.
3. An analysis of the applicability of LLM as a heuristic function for combinatorial graph optimization problems was conducted.

The thesis work is completed, the tasks set have been solved, there is the possibility of further development of the approach to other board games.

The thesis project was done solely by the author.

ВВЕДЕНИЕ

Объектом исследования данной дипломной работы является стохастическая игра с полной информацией. В качестве среды для тестирования алгоритмов выбрана настольная стратегическая игра немецкого стиля (Eurogame) Каркассон. Особенностями данной игры являются пошаговое выкладывание тайлов и четкие правила топологической связности (дорога к дороге, город к городу).

Актуальность исследования обусловлена следующими факторами: пространство состояний и возможных решений в данной игре исключает возможность полного перебора вариантов. Задача оптимального размещения тайлов является NP-трудной и эквивалентна обобщенной задаче Edge-Matching Puzzle. Кроме того, наличие отложенного вознаграждения (например, начисление очков за поля в конце игры) значительно снижает эффективность классических жадных алгоритмов.

Целью дипломной работы является разработка гибридного алгоритма оптимизации стратегии, объединяющего символьные вычисления (Model Context Protocol) и эвристический поиск на базе больших языковых моделей (LLM).

В ходе выполнения дипломной работы были поставлены следующие задачи:

- Описание игрового поля как динамического планарного графа, а размещения тайлов как задачи удовлетворения ограничений (CSP).
- Разработка игрового движка и сервера Model Context Protocol (MCP) для детерминированного расчета допустимости ходов и правил связности.
- Реализация интеграции и алгоритмов с использованием локальной LLM (Ollama) для генерации высокоуровневых стратегических гипотез.
- Проведение турнира агентов и сравнение гибридного подхода с существующими жадными алгоритмами.
- Оценка эффективности применения LLM в качестве эвристической функции.

ГЛАВА 1

Описание игры и целевая задача

1.1 Контекст и история развития настольных стратегий

Развитие интеллектуальных систем в области настольных игр прошло долгий путь от простейших эвристик для шашек до сверхмощных нейронных сетей для игры в Го [6]. Однако большинство классических исследований фокусировалось на играх с детерминированным процессом и полной информацией. Игра «Каркассон» (Carcassonne), созданная Клаусом-Юргеном Вреде в 2000 году [7], представляет собой принципиально другой класс задач.

«Каркассон» относится к так называемым играм «немецкого стиля» (Eurogames), где акцент смещен с прямого конфликта на оптимизацию ресурсов и долгосрочное планирование. Популярность игры и ее глубокие математические основы сделали ее «золотым стандартом» для тестирования современных алгоритмов искусственного интеллекта (ИИ), работающих в условиях неопределенности и стохастического выбора.

1.2 Архитектура и правила игрового процесса

Игровой мир «Каркассона» формируется динамически в реальном времени. В базовой версии игры участвуют два или более игроков, которые поочередно выстраивают ландшафт средневековой провинции, используя наборы квадратных тайлов.

1.2.1 Циклическая структура хода

Каждый игровой шаг формализован и строго разделен на три последовательные фазы:

1. **Фаза размещения тайла.** Игрок вытягивает случайный тайл из закрытой колоды. Тайл должен быть пристыкован к существующему полю по правилам

топологической консистентности. Стороны тайла (город, дорога, поле) должны логически продолжать объекты на соседних тайлах.

2. **Фаза размещения подданного (мипла).** Это ключевая стратегическая фаза. Игрок может выставить одну из своих ограниченных фишек (миплов) на один из объектов только что размещенного тайла. При этом объект не должен быть занят ранее.

3. **Фаза завершения и возврата.** Если действие привело к закрытию города, завершению дороги или окружению монастыря, происходит мгновенный подсчет очков. Использованные миплы возвращаются в запас игрока, что позволяет использовать их в последующих ходах.

1.2.2 Экономика ограниченных ресурсов

В отличие от многих стратегий, где ресурсы можно накапливать неограниченно, в Каркассоне игрок обладает всего 7 миплами. Это создает жесткое ограничение: «замораживание» мипла на крупном, но труднозавершаемом объекте (например, огромном строящемся городе) лишает игрока мобильности. Стратегическая задача ИИ состоит в поиске баланса между инвестициями в крупные объекты и поддержанием «ликвидности» запаса подданных.

1.3 Таксономия игровых стратегий и подсчета очков

Для разработки интеллектуального агента необходимо учитывать сложную систему весовых коэффициентов, связанных с различными типами ландшафта.

- **Дороги (Roads).** Самый простой вид объектов. Дают 1 очко за каждый тайл в составе завершенной дороги. Эффективны для быстрого возврата миплов.

- **Города (Cities).** Основной источник очков. Дает 2 очка за каждый тайл и 2 очка за каждый щит (эмблему) в составе города. Требуют сложного геометрического планирования для закрытия.

- **Монастыри (Cloisters).** Изолированные объекты. Завершаются, когда монастырь окружают 8 других тайлов. Дают фиксированные 9 очков.

- **Поля (Fields).** Наиболее сложная часть правил. Миплы на полях (крестьяне) не возвращаются до конца игры. Очки за них начисляются только в финальном финале (3 очка за каждый завершенный город, граничащий с полем). Это требует от алгоритма умения заглядывать за горизонт планирования на 50-70 ходов вперед.

1.4 Проблематика разработки интеллектуальных агентов

С точки зрения теории игр, Каркассон является стохастической игрой с неполной информацией (в части будущих тайлов колоды). Это порождает специфические трудности для автоматизации [3]:

1.4.1 Комбинаторный взрыв и branching factor

Количество допустимых позиций для тайла возрастает нелинейно. Если в начале игры вариантов всего 4-12, то к 30-му ходу количество вакантных мест на «границе» игрового поля может достигать 60. Учитывая 4 варианта поворота и 1-8 вариантов размещения мипла, количество ветвей в дереве поиска на один ход может превышать 500. Это делает невозможным применение традиционного исчерпывающего поиска (Brute Force).

1.4.2 Проблема «Действенного разрыва» (Action Gap)

Одной из центральных проблем, решаемых в данной работе, является разрыв между стратегическим видением («хочу построить город») и тактическим исполнением (какой из 20 доступных поворотов выбрать). Классические поисковые алгоритмы вроде MCTS [2] хорошо находят локальные максимумы, но плохо «понимают» концепцию блокировки противника или подготовки плацдарма для будущего города.

1.4.3 Баланс стохастики и детерминизма

Использование случайной колоды вносит элемент удачи. Эффективный агент должен не просто искать лучший ход, а максимизировать математическое ожида-

ние выгоды, учитывая вероятность выпадения нужного тайла в будущем. Это требует интеграции методов вероятностного анализа в структуру принятия решения.

1.5 Сравнительный анализ современных подходов

В современной практике разработки игрового ИИ выделяются три основных направления, каждое из которых имеет свои достоинства и недостатки:

1. **Эвристические методы (Heuristic Search).** Основаны на жестко заданных правилах оценки позиции. Примером является алгоритм Star2.5. Они быстры, но предсказуемы и лишены гибкости.

2. **Поиск по дереву Монте-Карло (MCTS).** Статистический метод, основанный на тысячах случайных симуляций партии (rollouts). MCTS доминирует в играх с глубоким деревом поиска, но в Каркассоне он часто упускает долгосрочные «полевые» стратегии из-за стохастического шума в симуляциях.

3. **Гибридные модели (Hybrid AI).** Сочетание классических алгоритмов с Large Language Models (LLM). Именно этот подход исследуется в данной работе, где LLM выступает в роли стратега («Генерала»), а поисковый движок — в роли тактического исполнителя («Солдата»).

Данная глава закладывает теоретический фундамент для последующей разработки и экспериментального сравнения указанных методов в рамках единой программной среды.

1.6 Сравнительный анализ с классическими игровыми дисциплинами

Для более глубокого понимания сложности Каркассона как задачи для ИИ, необходимо провести корреляцию с другими настольными играми, уже ставшими вехами в истории развития вычислительного интеллекта.

- **Carcassonne vs Chess.** В шахматах игровое поле фиксировано (8×8). Основная сложность заключается в глубине перебора вариантов. В Каркассоне само

поле является переменной величиной, что требует от алгоритма динамического перестроения графа связности на каждом шагу.

- **Carcassonne vs Go.** В Го количество состояний на доске 19×19 превосходит количество атомов во вселенной, однако информация на доске всегда полна. В Каркассоне присутствует «Скрытый фактор» (Blind deck) — неопределенность следующего тайла, что сближает его с карточными играми.

- **Carcassonne vs Poker.** Подобно Покеру, в Каркассоне критически важен учет вероятностей и риск-менеджмент. Однако, в отличие от покера, геометрическая составляющая (топология поля) накладывает на игрока жесткие логические ограничения, которые нельзя «заблефовать».

1.7 Психологические аспекты и «Мета-игра»

Цифровая реализация агента должна учитывать не только математическую выгоду, но и особенности человеческого восприятия. В партиях высокого уровня часто применяется стратегия «Психологического давления»:

- **Блокировка ключевых объектов.** Умышленное размещение тайла таким образом, чтобы противник никогда не смог достроить свой город (так называемые «мертвые города»). Для ИИ это требует понимания геометрии «невозможных стыковок».

- **Захват полей через экспансию.** Постепенное подведение своих крестьян к чужому крупному полю. Это форма скрытого противостояния, которую трудно выразить через мгновенную эвристику.

1.8 Постановка цели и задачи исследования

На основе проведенного анализа предметной области, целью данной магистерской диссертации является разработка и исследование гибридной архитектуры искусственного интеллекта для игры в Каркассон, сочетающей в себе точность традиционных методов поиска и стратегическую гибкость больших языковых моделей.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Спроектировать и реализовать программную платформу для моделирования игрового процесса с поддержкой Model Context Protocol (MCP).
2. Разработать гибридного агента, использующего концепцию «Генерал-Солдат» для разделения стратегического планирования и тактического хода.
3. Исследовать применимость методик Tree of Thoughts (ToT) и Reflexion для улучшения качества ходов LLM-агента без дообучения весов модели.
4. Провести серию экспериментальных турниров для сравнения эффективности гибридной модели с классическими алгоритмами (MCTS, Greedy, Heuristic).
5. Проанализировать собранные данные телеметрии и сформировать рекомендации по дальнейшей оптимизации стратегий ИИ в стохастических средах.

Решение этих задач позволит не только создать сильного игрового агента, но и внесет вклад в понимание того, как символьные и нейросетевые подходы могут эффективно взаимодействовать при решении сложных оптимизационных задач.

ГЛАВА 2

Математическая модель игры

2.1 Формализация игровых объектов и правил

Для перевода правил настольной игры в программную логику необходимо формализовать основные компоненты системы.

2.1.1 Тайл как информационный объект

Центральным объектом системы является тайл — квадрат местности, разделенный на функциональные зоны. Следуя подходу К. Хейден [3], каждый тайл описывается как структура, содержащая информацию о четырех своих сторонах (север, юг, запад, восток) и центральной области. Каждая сторона тайла имеет определенный тип (поле, город, дорога), что обеспечивает топологическую связность при стыковке: сторона S_1 тайла T_1 может быть соединена со стороной S_2 тайла T_2 только если их типы идентичны ($type(S_1) = type(S_2)$).

2.1.2 Графовое представление игрового поля

Для программной реализации игровое поле моделируется как динамический планарный граф $G = (V, E)$, где:

- V — множество вершин, представляющих собой элементарные сегменты тайлов.
- E — множество ребер, соединяющих сегменты одного типа как внутри тайла, так и на границах смежных тайлов.

Граф G является динамическим, так как на каждом ходу k в него добавляется подграф G_{tile} нового тайла, и строятся ребра E_{match}, связывающие его с существующей структурой поля.

При добавлении нового тайла на поле граф G модифицируется: добавляются новые вершины и строятся новые ребра, объединяющие смежные сегменты. Пример графового представления игрового поля представлен на рисунке 2.1.

Рисунок 2.1 Графовое представление игрового поля

2.2 Алгоритмическая реализация связности

Одной из ключевых задач игрового движка является отслеживание завершенности объектов (городов, дорог) и распределение очков между игроками.

2.2.1 Использование системы непересекающихся множеств (DSU)

Для эффективного управления компонентами связности в графе G применяется структура данных *Disjoint-Set Union* (DSU) с оптимизациями по рангу и сжатию путей. Каждый сегмент тайла принадлежит определенному множеству $S \in \mathcal{S}$. Операция $Find(v)$ позволяет определить, к какому логическому объекту (например, конкретному городу) принадлежит вершина v. Операция $Union(u, v)$ выпол-

няется при стыковке тайлов, объединяя сегменты в единый игровой объект.

Использование DSU позволяет решать следующие задачи за амортизированное время $O(\alpha(n))$ [4]:

- **Проверка завершенности:** Объект считается завершенным, если у всех входящих в него сегментов нет свободных (не пристыкованных) сторон.
- **Определение владельца:** Подсчет количества миплов каждого игрока в рамках одной компоненты связности.
- **Мгновенный пересчет очков:** Обновление игрового состояния сразу после добавления ребра в граф.

ГЛАВА 3

Разработка гибридного алгоритма оптимизации

3.1 Архитектурное разделение слоев

Архитектура программного комплекса построена на принципе полного разделения ответственности между правилами игры и стратегическим интеллектом. Это разделение реализовано через два ключевых уровня:

3.1.1 Логический уровень (Logic Layer)

Игровой движок, написанный на языке Python, инкапсулирует в себе:

- **Справочник тайлов:** Описание всех 72 плиток базового набора.
- **Геометрический валидатор:** Проверка допустимости координат и ориентации тайла.
- **Расчетный модуль:** Учет очков и мониторинг завершенности объектов через DSU.

Logic Layer функционирует как изолированный сервер, передающий информацию только по стандартизированным протоколам.

3.1.2 Стратегический уровень (Strategy Layer)

Верхнеуровневый агент, использующий локальную Large Language Model (LLM). Он не знает правил «в жестком коде», но получает описание текущей ситуации и доступные инструменты (tools) через интерфейс сервера.

3.2 Реализация Model Context Protocol (MCP)

Для взаимодействия слоев выбран протокол *Model Context Protocol* (MCP). Это позволяет превратить игровой движок в «умную периферию» для ИИ.

3.2.1 Компоненты MCP-сервера

Разработанный сервер предоставляет следующие абстракции:

- **Resources:** Текстовое описание текущего состояния поля и списка оставшихся тайлов в колоде.
- **Tools:** Функции `get_legal_moves` (возвращает список координат и поворотов), `place_tile` (совершает ход) и `get_score`.
- **Prompts:** Предустановленные шаблоны системных промптов, которые подготавливают LLM к роли профессионального игрока в Каркассон.

Взаимодействие осуществляется через JSON-RPC 2.0, что обеспечивает слабую связанность компонентов и позволяет в будущем легко заменить стратегический уровень (например, использовать другую модель LLM) без изменения кода игрового движка.

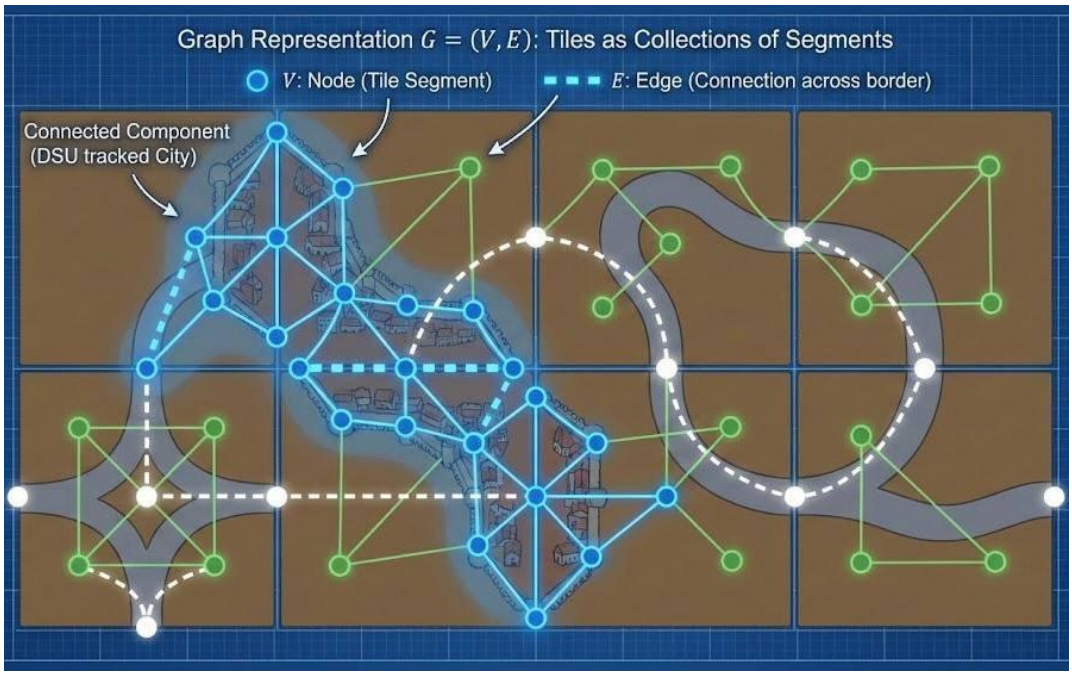

Рисунок 3.1 Схема архитектуры решения (Logic Layer и Strategy Layer)

3.3 Интеграция LLM и эвристики

В основе стратегического уровня лежит использование локальной языковой модели (Ollama), которая выступает в роли интеллектуального советника.

3.3.1 Методология Tree of Thoughts (ToT)

Для решения задачи планирования применяется фреймворк *Tree of Thoughts* [8]. В отличие от стандартного пошагового вывода, ToT позволяет модели:

- Генерировать несколько вариантов хода («мыслей»).
- Оценивать каждый вариант на предмет долгосрочной выгоды (Self-Evaluation).
- Выбирать наиболее перспективную ветвь развития или выполнять бэктрекинг в случае обнаружения тупиковых стратегий.

3.3.2 Вербальное подкрепление через Reflexion

Для минимизации ошибок и улучшения качества ходов без дорогостоящего дообучения модели внедрен подход *Reflexion* [5]. После каждого совершенного хода или в случае неудачного исхода партии, Logic Layer формирует текстовый отчет об ошибках, который передается модели. Модель выполняет «вербальную рефлексию», записывая выводы в оперативную память контекста, что повышает точность последующих решений.

3.4 Гибридный поиск и оптимизация

Хотя LLM способна к высокоуровневому планированию, она может допускать ошибки в точных расчетах. Для компенсации этого разработана гибридная модель:

- **MCTS как фильтр:** Традиционный Monte Carlo Tree Search [1] выполняет быструю оценку терминальных состояний для отсечения заведомо проигрышных ходов.
- **LLM как селектор:** Языковая модель выбирает финальный ход из 3-5 лучших вариантов, предложенных MCTS, основываясь на стратегическом контексте, который трудно формализовать в чистой математике (например, психологическое давление на оппонента).

Такой симбиоз позволяет достичь баланса между вычислительной точностью и стратегической гибкостью.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была решена задача создания автономного игрового агента для стохастической настольной игры с полной информацией Каркассон.

Разработанный программный комплекс состоит из двух интегрированных слоев: детерминированного математического ядра (Logic Layer), реализованного в виде сервера Model Context Protocol, и эвристического модуля генерации стратегий (Strategy Layer), работающего на базе локальной языковой модели Ollama.

Полученные результаты демонстрируют принципиальную возможность использования современных больших языковых моделей (LLM) для решения NP-трудных задач комбинаторной оптимизации на динамических планарных графах. Архитектура взаимодействия посредством вызова инструментов (Tool Use / MCP) позволила компенсировать известную проблему галлюцинаций LLM и делегировать строгую проверку топологической связности и подсчет очков четкому алгоритмическому движку.

В качестве дальнейшего развития разработанного подхода планируется расширение эвристик агента за счет интеграции методов подкрепленного обучения (Reinforcement Learning) и алгоритмов поиска по дереву состояний (Monte Carlo Tree Search), а также адаптация обобщенного математического ядра для других настольных игр со схожей механикой размещения.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] F. Ameneyro et al. Playing carcassonne with monte carlo tree search. 2020.

[2] C. B. Browne et al. A survey of monte carlo tree search methods. In *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[3] C. Heyden. Implementing a computer player for carcassonne. (star2.5 analysis). 2009.

[4] D. E. Knuth. Dancing links. (exact cover algorithms).

[5] N. Shinn et al. Reflexion: Language agents with verbal reinforcement learning. (Методология обучения через вербальную рефлексию). 2023.

[6] D. Silver et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm (alphazero). *arXiv preprint arXiv:1712.01815*, 2017.

[7] K.-J. Wrede. *Carcassonne: The Rulebook and Concept*. Hans im Glück, 2000.

[8] S. Yao et al. Tree of thoughts: Deliberate problem solving with large language models. (Методология использования llm для стратегического планирования). 2024.

ПРИЛОЖЕНИЕ А

Репозиторий проекта на GitHub

Репозиторий проекта на GitHub