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 Правила и особенности игры Каркассон

Игра «Каркассон» (Carcassonne) является настольной стратегической игрой немецкого стиля (Eurogame), в которой игровой процесс строится на пошаговом выкладывании квадратов местности (тайлов) и размещении на них фишек подданных (миплов) [4]. Пример игрового процесса представлен на рисунке 1.1.

Рисунок 1.1 Игровой процесс и выкладывание тайлов в Каркассоне

Игровое пространство формируется динамически в процессе игры. Каждый

новый тайл должен быть выложен по правилам топологической связности: грани выкладываемого квадрата должны логически продолжать объекты на уже лежащих на столе квадратах (дорога должна соединяться с дорогой, город с городом, поле с полем).

1.2 Проблематика алгоритмического решения

С точки зрения теории игр, Каркассон представляет собой стохастическую игру с полной информацией. Однако разработка оптимальной стратегии для искусственного агента сталкивается с рядом существенных трудностей:

- **Размер пространства состояний.** В базовой версии игры участвуют 72 тайла, которые могут быть выложены в различных конфигурациях. Количество допустимых позиций возрастает экспоненциально с каждым ходом [2].
- **NP-трудность.** Задача поиска оптимального размещения тайлов математически эквивалентна обобщенной задаче подбора краев (Edge-Matching Puzzle), которая в общем случае является NP-полной.
- **Проблема горизонта планирования.** Из-за механизма подсчета очков (в частности, отложенного вознаграждения за владение полями в самом конце игры) классические жадные алгоритмы (Greedy algorithms) принимают локально оптимальные, но глобально убыточные решения [6].

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 [2], 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 $O(\alpha(n))$ [3]:

- **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.

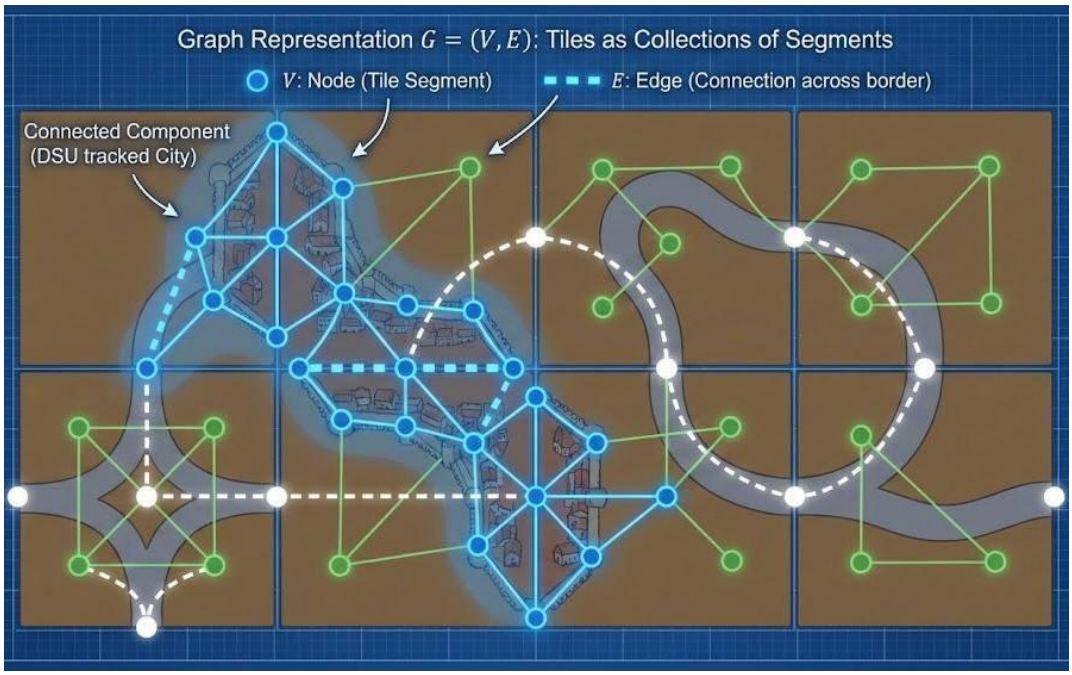

Рисунок 3.1 Схема архитектуры решения (Logic Layer и Strategy Layer)

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

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

[4] L. Kárná. Carcassonne – description of the game. (theory of graphs application), 2012.

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

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction. (Теория стохастических сред и отложенного вознаграждения)*. MIT press, 2018.

[7] 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