МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ПРИМЕНЕНИЕ РОЕВЫХ АЛГОРИТМОВ ДЛЯ ПОИСКА КРАТЧАЙШЕГО ПУТИ В ЛАБИРИНТЕ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Доронкина Михаила Александровича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

ВВЕДЕНИЕ

В современном мире проблема поиска кратчайшего пути становится всё более актуальной. Города растут, с ними и дороги, появляются всё больше улиц и для нахождения более коротких путей/актуальных между определенными точка на карте становится всё сложнее. Для решения задач существует множество алгоритмов и идей реализации, но у каждых есть свои определенные минусы, от количество затраченной памяти на поиск решения, до времени выполнения, что в свою очередь довольно актуально, так же часто требуется учитывать какие-то внешние факторы и множество ещё разных факторов. Роевые алгоритмы являются одним из наиболее перспективных методов оптимизации нахождения пути, которые находят широкое применение в различных областях, включая информационные технологии, транспорт, производство и многие другие.

Лабиринты - это увлекательная, захватывающая структура, которую за частую бывает очень сложно решить. Транспортные сети, социальные сети все это можно безукоризненно назвать разновидностью лабиринтов. В таких лабиринтов зачастую бывает не одно единственно решение, а множество, для решения этой задачи используются различные алгоритмы, среди которых особое место занимают роевые алгоритмы.

Роевые алгоритмы основаны на принципах коллективного поведения живых организмов, таких как муравьи, пчёлы и рыбы. Эти алгоритмы позволяют находить оптимальные решения сложных задач, используя минимальное количество ресурсов и времени.

Целью данной данной работы является исследование возможности применения роевых алгоритмов для поиска кратчайшего пути в лабиринтах. В работе будут рассмотрены основные принципы работы роевых алгоритмов, а также проведены эксперименты по сравнению эффективности с другими алгоритмом, таким как волновой алгоритм.

В данной работе поставлены следующие задачи:

- генерация лабиринта со множеством путей;
- рассмотрение основных принципов роевого интеллекта, реализация алгоритма муравьиной колонии и алгоритма формирования реки;
- реализация волнового алгоритма и сравнение его эффективности с роевыми алгоритмами по времени и длине находимого кратчайшего пути;
- оценка эффективности применения роевых алгоритмов для поиска крат-

чайшего пути в лабиринтах.

— разработка графического интерфейса для каждого из этапов.

Данная работа состоит из введения, трех глав и заключения. В первой главе дается определение различным видам лабиринтов, описываются способы их генерации и прохождения. Во второй главе рассматривается понятие роевого интеллекта, детально описываются Алгоритм муравьиной колонии и Алгоритм формирования реки. В третьей главе описываются этапы создания программы и производится сравнительный анализ эффективности алгоритмов поиска кратчайшего пути в лабиринте.

1 Лабиринты

Лабиринт можно рассматривать как некую структуру, отображенную в двухмерном или трехмерном пространствах, характерной чертой которой является наличие запутанных путей, ведущих от входа к выходу или к многочисленным тупикам. Во время античности лабиринты отождествлялись с мифами о Кносском лабиринте на Крите, где томился Минотавр. Сейчас же значение и применение данной структуры куда шире.

1.1 Графы

Лабиринт можно рассматривать как частный случай графа, где вершины представляют собой перекрёстки или конечные точки, а рёбра — коридоры между ними. В контексте теории графов, лабиринт — это граф, где:

- Каждая локация является вершиной.
- Каждый проход между локациями является ребром.
- Каждая пара локаций соединена как минимум одним проходом.

Формализация лабиринта в терминах теории графов позволяет классифицировать его как определённый класс графов. Идеальный лабиринт характеризуется как:

- Неориентированный: рёбра не имеют направления.
- Несвязный: не все вершины соединены напрямую.
- Простой: отсутствуют петли и кратные рёбра.

Таким образом, задача создания лабиринта сводится к задаче генерации графа с определёнными характеристиками. Это может быть упрощено до задачи создания минимального остовного дерева, где веса рёбер не учитываются, поскольку в определении лабиринта не задана стоимость прохода между локациями. Проблема поиска пути в лабиринте также сводится к задаче нахождения пути между двумя вершинами в графе.

1.2 Генерация лабиринтов

Совокупности поэтапных правил, позволяющих создать определенные заданные структуры лабиринтов, можно назвать алгоритмами генерации лабиринтов. Генерация лабиринтов может найти себе прекрасное применение на практике в качестве ресурса для добавления усложненности в проекты разного рода. Самым банальным примером можно счесть целый жанр игр, завязанных на прохождении лабиринтов. Помимо этого генерация лабиринтов обладает целым

рядом возможностей, включающих такие, как:

- Создание ландшафта с набором препятствий
- Создание планировки подземелий
- Генерация пазлов
- Создание различных визуальных эффектов
- Обучение алгоритмам графов

На сегодняшний день существует целый ряд алгоритмов, позволяющих сгенерировать достаточно сложные и разнообразные лабиринты. К таким алгоритмам можно отнести:

- Алгоритм «рекурсивного поиска с возвратом» - довольно быстр, легок для понимания и прост в реализации. Однако является требовательным к наличию памяти, из-за чего требует использования пространства стека, пропорционального размерам итогового лабиринта, что не является эффективным решением при построении исключительно больших лабиринтов
- Алгоритм Эйлера — один из самых специфичных, но также один из самых быстрых. Данный алгоритм позволяет создать лабиринт любого размера за линейное время.Это делается путем построения лабиринта по одной строке за раз, помещая ячейки этих строк в наборы для отслеживания того, какие столбцы в конечном итоге соединены. Алгоритму нет необходимости просматривать более одной строки, и когда процесс подходит к концу, он всегда создает идеальный лабиринт.
- Алгоритм «рекурсивного деления» должен быть реализован как сумматор стен. Этот алгоритм особенно интересен из-за своей фрактальной природы: В теории, можно продолжать процесс бесконечно, постепенно увеличивая уровень детализации.

1.3 Алгоритмы поиска путей в лабиринтах

Алгоритмы поиска путей в лабиринтах могут быть основаны на различных математических структурах. Например, алгоритмы, использующие графы, моделируют сеть тонких путей и вершин, через которые можно проходить. Двумерные массивы позволяют создать сетку, на которой можно отслеживать перемещение и искать кратчайшие пути.

- Алгоритм одной руки: перемещаясь по лабиринту, необходимо постоянно идти вдоль правой или левой из его стен. Вероятно, этот метод был ши-

роко известен еще в древности, возможно, в древней Греции. Хоть этот подход может привести к долгому пути с преодолением всех тупиков, в конечном итоге он помогает достичь желаемой цели. Алгоритм одной руки, также известный как «правило левой/правой руки», гарантированно находит выход из односвязного лабиринта. Поскольку все стены соединены, в конечном итоге человек достигнет выхода.

— Алгоритм Тремо, разработанный Шарлем Пьером Тремо, представляет собой метод решения лабиринта, который использует рисование линий и точек по мере прохождения для обозначения пути. Этот подход позволяет решать как простые, так и многосвязные лабиринты. Существуют определенные правила, которые следует соблюдать при использовании алгоритма Тремо, чтобы успешно найти выход из лабиринта. Такой метод не только помогает найти путь, но и визуализирует этот путь на лабиринте, облегчая понимание процесса поиска решения.
— Волновой алгоритм — это алгоритм графового поиска, который используется для нахождения всех кратчайших путей от одной начальной вершины до всех остальных вершин в графе. Он работает, распространяя волны из начальной вершины во всех направлениях, пока не достигнет всех вершин графа.

2 Роевой интеллект

Роевой интеллект возникает, когда группа существ, обладающих ограниченными когнитивными способностями, объединяется для выполнения сложных задач. Хотя действия отдельных существ могут быть простыми и непредсказуемыми, коллективное поведение группы демонстрирует поразительную сложность и эффективность.

Ключевым фактором в роевом интеллекте является наличие простых правил, которым следуют все члены группы. Эти правила определяют, как существа взаимодействуют друг с другом и с окружающей средой. Например, пчелы используют танец, чтобы сообщать другим членам улья о местонахождении источника пищи. Муравьи используют феромоны, чтобы прокладывать тропы к источникам пищи и к гнезду.

Несмотря на то, что отдельные существа могут вести себя хаотично, коллективное поведение роя определяется следующими принципами:

- Разнообразие: рой хранит ресурсы во многих местах, а не в одном центральном хранилище.
- Адаптивность: рой может изменять свое поведение в ответ на изменения окружающей среды.
- Приближение: рой может проводить простые вычисления для оценки времени и пространства.
- Стабильность: рой поддерживает постоянство своего поведения, даже после изменений окружающей среды.
- Качество: рой реагирует на факторы качества окружающей среды, такие как безопасность, наличие пищи и другие.

Благодаря этим принципам рои могут эффективно решать сложные проблемы, такие как навигация, поиск пищи и оптимизация использования ресурсов. Роевой интеллект находит применение в различных областях, включая робототехнику, оптимизацию и искусственный интеллект.

2.1 Принцип работы роевых алгоритмов

Концепция роевого интеллекта использует рой частиц, где каждая частица представляет потенциальное решение проблемы. Поведение частицы в гиперпространстве поиска решения все время подстраивается в соответствии со своим опытом и опытом своих соседей. Кроме этого, каждая частица помнит

свою лучшую позицию с достигнутым локальным лучшим значением целевой функции и знает наилучшую позицию частиц - своих соседей, где достигнут глобальный на текущий момент оптимум. В процессе поиска частицы роя обмениваются информацией о достигнутых лучших результатах и изменяют свои позиции и скорости по определенным правилам на основе имеющейся на текущий момент информации о локальных и глобальных достижениях. При этом глобальный лучший результат известен всем частицам и немедленно корректируется в том случае, когда некоторая частица роя находит лучшую позицию с результатом, превосходящим текущий глобальный оптимум. Каждая частица сохраняет значения координат своей траектории с соответствующими лучшими значениями целевой функции.

2.2 Муравьиный алгоритм

Муравьиный алгоритм является примером коллективного интеллекта, где децентрализованные агенты (муравьи) действуют локально, но вместе способны достичь глобальной оптимизации. Поведение муравьев основано на простых правилах - они следуют путем, где сильнее концентрация феромонов, что позволяет им обходить препятствия и находить оптимальные маршруты. По мере итераций в процессе поиска оптимального решения, муравьиный алгоритм способен находить более эффективные и улучшенные решения благодаря комбинированию индивидуальных усилий каждого муравья в колонии.Этот метод широко применяется в различных областях, где необходимо решать задачи оптимизации, такие как маршрутизация сетей, планирование производства, раскрой материалов и другие.

Работу алгоритма можно описать следующем образом:

1. Инициализация: создаётся виртуальная колония муравьёв, количество которых обычно соответствует числу точек в решаемой задаче.
2. Поиск: муравьи отправляются на поиск решения задачи. Каждый муравей строит решение, последовательно посещая точки в случайном порядке до тех пор, пока не вернется в исходную точку или не окажется в тупике
3. Оставление следа: после завершения поиска каждый муравей оставляет след из феромонов вдоль пути, по которому он двигался. Количество оставленных феромонов пропорционально качеству решения, найденного муравьем.
4. Выбор лучшего пути: муравьи с большей вероятностью будут следовать

путям, которые имеют более сильный след феромонов. Это приводит к положительной обратной связи, при которой хорошие решения еще больше усиливаются, а плохие решения ослабевают.

5. Обновление следа: после каждого цикла поиска след феромонов обновляется для учета недавно найденных решений. Феромоны постепенно испаряются с течением времени, уменьшая влияние старых решений и давая возможность новым решениям быть обнаруженными.
6. Повторение: процесс поиска и обновления следа повторяется заданное количество раз или до тех пор, пока не будет найдено удовлетворительное решение.

2.3 Алгоритм формирования рек

Алгоритм формирования рек (англ. river formation dynamics) был предложен группой ученых (англ.) P. Rabanal, I. Rodriguez, и F. Rubio в 2007 году как альтернатива алгоритму муравьиной колонии для поиска оптимального решения задачи коммивояжера и других NP-полных задах. Недостатком муравьиного алгоритма они считают возможность того, что в случае нахождения лучшего пути, иногда требуется очень большое количество итераций для его укрепления, пока он не будет более привлекательным, чем уже укрепленные пути.

Краткое описание алгоритма RFD выглядит следующим образом:

1. Вначале среда плоская, т.е. высота всех вершин одинакова, за исключением целевой вершины, которая равна нулю в течение всего процесса.
2. Капли, двигаясь, размывают свои траектории (забирая часть почвы из вершин) или осаждают переносимый осадок (таким образом, увеличивая высоту вершин). Вероятность выбора следующей вершины зависит от градиента, который пропорционален разнице между высотой вершины, на котором находится местоположение капли и высота ее соседа.
3. Капли помещаются в начальный узел, чтобы обеспечить возможность дальнейшего изучения окружающей среды.
4. На каждом шаге группа капель последовательно пересекает пространство, а затем выполняет эрозию на посещенных вершинах.

3 Реализация программы

Данная работа был выполнена в «WindowsForm» на языке программирования C++ на операционной системе Windows 10 с процессором 11th Gen Intel(R) Core(TM) i7-11800H 2.30GHz 2.30 GHz 16GB RAM.

Для упрощенного хранении/передачи лабиринтам между алгоритмами поиска пути, был реализован свой собственный граф, на основе словаря.

Для наглядности кроме реализации «Муравьиной колонии» и «Формирования рек» был реализован алгоритм «Формирования рек»

По завершению работы алгоритма, программа выводит:

— размер лабиринта. Размер лабиринта рассчитывается путём умножения количества строк на количество столбцов.
— время работы каждого алгоритма.
— кратчайший путь найденный алгоритмом, обозначенный зеленым цветом.
— так же серым, для роевых алгоритмов показаны пути которые прошли агенты.

Также отображается сгенерированный лабиринт в виде изображения (см. рисунок 1), поверх которого выводится изображение пути после применения муравьиного алгоритма и найденного им кратчайшего пути. Так же, может быть представлено изображение лабиринта после применения алгоритма формирования рек и найденного им кратчайшего пути. Ещё одним вариантом является использование волнового алгоритма для поиска кратчайшего пути, в зависимости от выбора (см. рисунок 2).

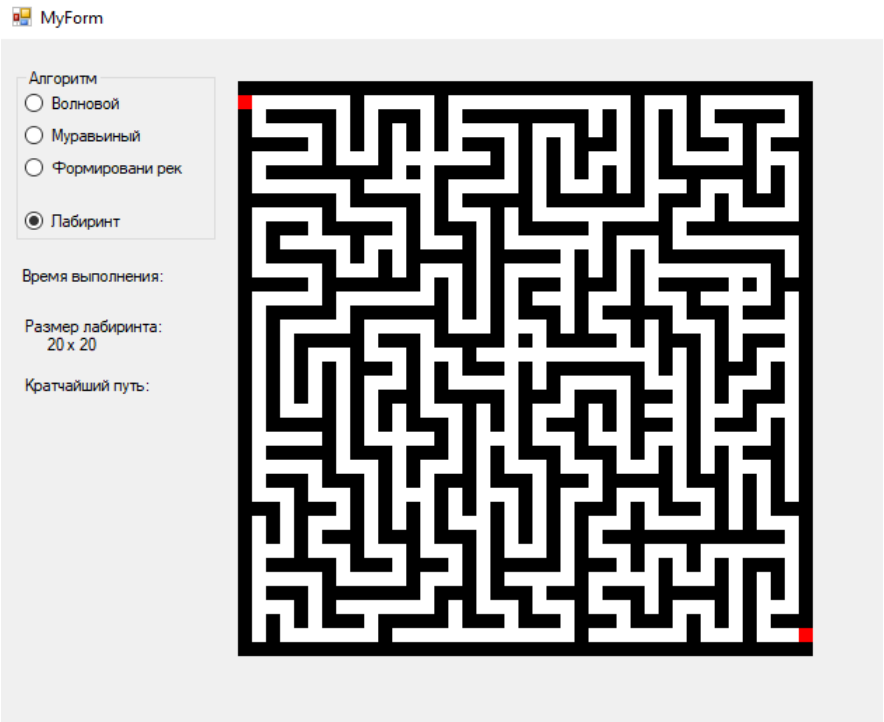

Рисунок 1 – Пример работы формы без выбора алгоритма

Рисунок 2 – Пример работы формы при выбранном алгоритме

3.1 Генерация лабиринта со множеством путей

Генерация лабиринта задача довольно сложная, но как и присуще такой интересной задаче существует множество алгоритмов для решения поставленной задачи. Но сколько существует алгоритмов столько существует и идей фор-

мирования лабиринта с множественными путями. В основном в алгоритмах для формирования лабиринта с множественными путями лежат модифицированные алгоритма формирования идеально лабиринта.

За основу в данной работе был взят алгоритм алгоритм «поиска с возвратом», так же его называют алгоритмом на графах «поиск в глубину» - алгоритм достаточно прост в понимании и в написании.

3.2 Муравьиный алгоритм

Основная идея муравьиного алгоритма лежит над сильно-связным графом, что в свою очередь не подходил для лабиринта, поэтому его необходимо немного модифицировать.

Первое что необходимо учитывать что хоть и лабиринт можно представить как граф, он не будет являть сильно-связным. Наличие тупиков, когда можно упереться в стену следуя определенному пути, тоже необходимо учитывать. Поэтому, необходимо таких ситуация разрешить муравью возвращаться. Если муравей попал в тупик, то ему необходимо будет вернуть к развилке, где сможет выбрать другой путь, опустошив феромоны за собою.

Второе что необходимо учитывать, что лабиринт это граф, но длина/вес пути между вершина везде одинакова, и равна единице.

3.3 Алгоритм формирования рек

Алгоритм хорошо себя показывает и без необходимости внесения изменений в исходный код, несмотря на наличие большого количества статических данных, требующих ручной настройки. Тем не менее, для улучшения его производительности можно внести определенные модификации.

Как и в случае с алгоритмом, основанным на поведении муравьев, важно учитывать возможность возникновения тупиковых ситуаций в лабиринте. Несмотря на то, что алгоритм изначально предусматривает такие сценарии, при попадании капли в тупик она испаряется и при этом всё равно откладывается часть переносимого осадка, что отличает данный подход от поведения муравьев, где феромоны оставляются только теми особями, которые достигли конечной точки. Однако, это может занять значительное количество времени и ресурсов когда капля сможет пройти от начала лабиринта и до его конца. Поэтому, в таком случаи разрешаем капле вернуться обратно, но на таких вершина вместо откладывания части переносимого осадка, сразу добавляем некоторое

количество земли.

3.4 Сравнение алгоритмов

Сравнительный анализ показал, что Алгоритм муравьиной колонии (АМК) и Алгоритм формирования реки (АФР) достигают практически одинаковых результатов по длине находимого кратчайшего пути в лабиринтах различных размеров. Для лабиринтов небольших размеров скорость их работы также отличается незначительно, но для лабиринтов площадью 900 вершин и более, АФР затрачивает времени в два раза меньше, чем АМК. Для лабиринтов площадью более 10000 вершин, скорость их работы начинается отличаться приблизительно в 5 раз, из чего можно сделать вывод, что АФР является менее эффективным роевым алгоритмом для поиска кратчайшего пути. В сравнении с алгоритмам роевого интеллекта, Волновой алгоритм показывает себя очень хорошо, время и путь всегда минимален. Его главным преимуществом является быстрота работы даже в лабиринтах больших размеров, поскольку затрачиваемое время для поиска кратчайшего пути составляет менее секунды, даже в лабиринтах площадью более 10000 вершин.

ЗАКЛЮЧЕНИЕ

Все алгоритмы, рассмотренные выше, были сравнены по времени работы и длине находимого ими кратчайшего пути в лабиринте.

Для каждого размера лабиринта было найдено среднее арифметическое результатов трех опытов (с округлением в большую сторону длины кратчайшего пути).Сравнительный анализ показал, что Алгоритм муравьиной колонии (АМК) и Алгоритм формирования реки (АФР) достигают практически одинаковых результатов по длине находимого кратчайшего пути в лабиринтах различных размеров. Для лабиринтов небольших размеров скорость их работы также отличается незначительно, но для лабиринтов площадью 900 вершин и более, АМК затрачивает времени в два раза меньше, чем АФР. Для лабиринтов площадью более 10000 вершин, скорость их работы начинается отличаться приблизительно в 5 раз, из чего можно сделать вывод, что АМК является более эффективным роевым алгоритмом для поиска кратчайшего пути. В сравнении с алгоритмам роевого интеллекта, Волновой алгоритм показывает самые лучшие результаты по длине находимого кратчайшего пути. И его главное преимуществом является быстрота работы даже в лабиринтах больших размеров, поскольку затрачиваемое время для поиска кратчайшего пути составляет менее секунды, даже в лабиринтах площадью более 10000 вершин.