Министерство образования и науки Российской Федерации

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

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Калужской Алины Сергеевны

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

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

Саратов 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Удивительно слаженное поведение многих роевых животных всегда вызывало интерес человека. Возникал вопрос — как такое возможно без системы планирования и централизованного управления. Такое поведение живых организмов получило название — «коллективный разум», а искуственное моделирование его человеком для решения определенных задач — алгоритмами «роевого интеллекта». Системы роевого интеллекта состоят из множества живых организмов (агентов), локально взаимодействующих между собой и с окружающей средой.

В настоящее время лабиринты часто используются для моделирования различных ситуаций. Например, поиск оптимального направления движения во время транспортного затора или прокладывание соединений между узлами связи. Проблемы оптимизации в таких моделях нередко могут быть сформулированы в термах поиска путей в лабиринте. Существует множество алгоритмов прохождения лабиринтов и поиска кратчайших путей в них, в данной работе исследуется возможность применения роевых алгоритмов для этой проблемы.

Целями данной работы являются:

- генерация лабиринта со множеством путей
- изучение 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 кратчайшего 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

Первоначальный смысл слова «лабиринт» подразумевает только извилистую тропу. Сейчас под этим словом понимается не только тропа, но и набор дорожек, которые могут закручиваться, пересекаться, иметь витки или вести в тупик. Поэтому лабиринт представляет собой множество путей, ведущих от начальной точки до выхода, которые отделены друг от друга непроницаемыми стенами [1].

1.1 Алгоритмы поиска путей в лабиринтах

— Алгоритм одной руки. Является наиболее известным алгоритмом поиска путей в лабиринте. Также известен как «правило левой руки» или «правило правой руки». Если лабиринт является односвязным, то есть все его стены соединены между собой или с внешней границей лабиринта, значит правило одной руки гарантированно применимо к нему, и не отрывая одной из рук на каждом повороте, игрок обязательно придет к выходу (или к начальной точке, если выхода не существует). Недостатком данного алгоритма является его неприменимость к многосвязным сложным лабиринтам, поскольку в них часто присутствуют циклы, что делает невозможным поиск кратчайшего пути с помощью данного алгоритма.

— Алгоритм Пледжа. Разновидностью алгоритма одной руки является алгоритм Пледжа, используя который игрок выбирает определенное направление и движется вперед насколько это возможно. Если на его пути оказывается стена, он применяет правило одной руки, пока первоначальное направление снова не будет доступно. Применяя правило одной руки, игрок считает количество сделанных поворотов, например поворот налево как -1, а направо как +1. Игрок перестает использовать правило одной руки, только если общая сумма совершенных поворотов равна 0. Подсчет гарантирует, что в конечном итоге игрок продвинется вперед по лабиринту (а не повернет назад на 360 градусов), и перейдет к следующему множеству стен, одно из которых будет соединено с выходом [2].

— Алгоритм Tremaux. Данный алгоритм гарантированно работает даже в многосвязных лабиринтах. Каждая дорожка лабиринта может быть непосещенной, отмеченной один раз или отмеченной дважды. В начале пути

игрок выбирает произвольное направление движения. Каждый раз выбирая направление, игрок рисует линию на полу (от одного пересечения дорожек до другого). При попадании на еще не посещенный перекресток, игрок выбирает любую дорожку и помечает ее. Если игрок оказывается на помеченном перекрестке, а его дорожка имеет только одну линию, он разворачивается и идет назад, рисуя на ней вторую линию. Если игрок добирается до финиша, дорожки, имеющие только одну линию, будут является прямым путем к начальной клетке. А иначе путь назад будет состоять только из дорожек, помеченных дважды. Поиск такого пути называется двунаправленной двойной трассировкой [3].

1.2 Волновой алгоритм

Волновой алгоритм (алгоритм волновой трассировки, алгоритм Ли) — алгоритм поиска пути, алгоритм поиска кратчайшего пути на планарном графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину. Волновой алгоритм в контексте поиска пути в лабиринте был предложен Э. Ф. Муром. Ли независимо открыл этот же алгоритм при формализации алгоритмов трассировки печатных плат в 1961 году. [4]

Описание алгоритма:

1. Пометить стартовую ячейку $d = 0$
2. Для каждой ячейки, помеченной числом d, пометить все соседние свободные непомеченные ячейки числом $d + 1$, ПОКА (финишная ячейка не помечена) И (есть возможность распространения волны)
3. ЕСЛИ финишная ячейка помечена, ТО перейти в финишную ячейку
4. Выбрать среди соседних ячейку, помеченную числом на 1 меньше числа в текущей ячейке, перейти в выбранную ячейку и добавить её к пути, ПОКА текущая ячейка — не стартовая

2 Роевой интеллект

Под алгоритмами роевого интеллекта понимается класс алгоритмов, направленных на решение сложных оптимизационных задач (дискретная оптимизация, многомерная оптимизация и многокритериальная оптимизация), работа которых основана на моделировании коллективного поведения различных колоний живых организмов. Биологические и социальные группы живых организмов часто демонстрируют особый вид общения и интеллекта, которая возникает из-за децентрализованное принятия решений и индивидуального общения между организмами. Системы роевого интеллекта состоят из организмов, где каждый участник является равноправным, которые взаимодействуют между друг другом и окружающей средой, но только на «низком уровне», то есть не согласуя свои действия с центральной системой принятия решений. Поэтому роевые алгоритмы представляют особый класс интеллекта, зачастую, сильно отличающийся от человеческого. Они эффективно применяются в случаях с ограниченным или недостаточным количеством знаний о проблеме [5].

2.1 Характеристики роевых систем

Типичная система роевого интеллекта обладает следующими свойствами:

- Она состоит из большого количества участников.
- Все участники практически идентичны (принадлежат к одному и тому же или очень схожим биологическим видам).
- Взаимодействия между участниками подчиняются простым поведенческим правилам, использующим только локальную информацию, которой они обмениваются непосредственно или через окружающую среду (стигмергия).
- Общее изменение всей системы происходит непосредственно как результат взаимодействия индивидов друг с другом и с окружающей средой.

2.2 Алгоритм муравьиной колонии

Муравьиный алгоритм (алгоритм оптимизации подражанием муравьиной колонии, англ. ant colony optimization, ACO) — один из эффективных полиномиальных алгоритмов для нахождения приближенных решений задачи коммивояжера, а также решения аналогичных задач поиска маршрутов на

графах. Суть подхода заключается в анализе и использовании модели поведения муравьев, ищущих пути от колонии к источнику питания и представляет собой метаэвристическую оптимизацию. Первая версия алгоритма, предложенная доктором наук Марко Дориго в 1992 году, была направлена на поиск оптимального пути в графе [6].

В реальном мире, муравьи (первоначально) ходят в случайном порядке и по нахождению продовольствия возвращаются в свою колонию, прокладывая феромонами тропы. Если другие муравьи находят такие тропы, они, вероятнее всего, пойдут по ним. Вместо того, чтобы отслеживать цепочку, они укрепляют её при возвращении, если в конечном итоге находят источник питания. Со временем феромонная тропа начинает испаряться, тем самым уменьшая свою привлекательную силу. Чем больше времени требуется для прохождения пути до цели и обратно, тем сильнее испарится феромонная тропа. На коротком пути, для сравнения, прохождение будет более быстрым и как следствие, плотность феромонов остается высокой. Испарение феромонов также имеет свойство избежания стремления к локально-оптимальному решению. Если бы феромоны не испарялись, то путь, выбранный первым, был бы самым привлекательным. В этом случае, исследования пространственных решений были бы ограниченными. Таким образом, когда один муравей находит (например, короткий) путь от колонии до источника пищи, другие муравьи, скорее всего пойдут по этому пути, и положительные отзывы в конечном итоге приводят всех муравьев к одному, кратчайшему, пути.

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.

Данный алгоритм иммитирует формирование реки на изначально ровной местности. Потоки рек размывают почву под ними, и поднимают седименты (седименты, наносы — твёрдые частицы, переносимые водным или воздуш-

ным потоком) с земли. Потоки несут их дальше, пока они не осядут ниже по течению реки по ее сторонам. Полагая, что реки всегда текут от возвышенностей к низинам, места с большим количеством седимента становятся менее благоприятными для следующих потоков, что уменьшает их вероятность быть выбранными на пересечениях. В свою очередь уже размытые пути с меньшим количеством седимента будут выбраны с большей вероятностью, поскольку они позволяют реке течь быстрее, что в последствии еще больше размывает эти пути.

В случае применения данного алгоритма для поиска кратчайшего пути в лабиринте, стартовая клетка представляется в виде непрерывного источника воды, а финишная клетка в виде ямы (в которой не осаждаются седименты и не остается вода). Потоки воды проходят по лабиринту, изменяют уровень почвы, и размывают пути. Уровень почвы и количество седиментов зависит от длины найденного пути. По окончанию всех итераций, в лабиринт запускается тестовый поток, который более не меняет уровень почвы, а выбирает наиболее предпочтительный путь на каждом перекрестке. Этот поток и определяет кратчайший путь [7].

Для поиска кратчайшего пути в лабиринте алгоритм был модифицирован следующим образом (все значения были подобраны экспериментально):

3 Разработка программы

В данной работе используется язык программирования Python версии 2.7 на операционной системе Windows 10 с процессором Intel Core $i3$ 2.30 GHz 4GB RAM.

Пример созданного лабиринта с длиной и шириной 51×51 клеток можно увидеть на рисунке 1.

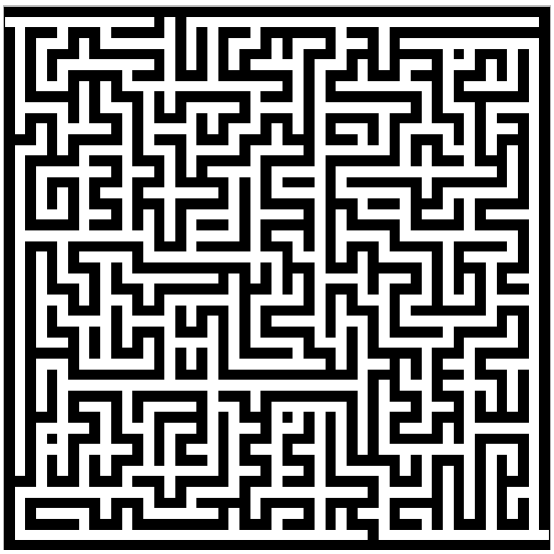

Рисунок 1 – Созданный лабиринт

3.1 Поиск кратчайшего пути

В данной работе поиск кратчайшего пути в лабиринте был реализован с помощью таких роевых алгоритмов как Алгоритм муравьиной колонии и Алгоритм формирования рек, а также, для сравнение, с помощью Волнового алгоритма — известного алгоритма для поиска кратчайшего пути в лабиринтах и графах, не относящегося к роевому интеллекту.

По завершению работы, программа выводит размер лабиринта, время работы каждого алгоритма, и кратчайший путь, найденным ими. Также отображаются пять графиков: матрица лабиринта после применения Муравьиного алгоритма и найденный им кратчайший путь, матрица лабиринта после применения Алгоритма формирования рек и найденный им кратчайший путь, и найденный кратчайший путь с помощью Волнового алгоритма.

Пример работы программы для лабиринта 35×35 клеток:

— Результат работы программы (рис.2).
— Кратчайший путь, найденный Муравьиным алгоритмом (рис.3).
— Кратчайший путь, найденный Алгоритмом формирования рек (рис.4).

— Кратчайший путь, найденный Волновым алгоритмом (рис.5).

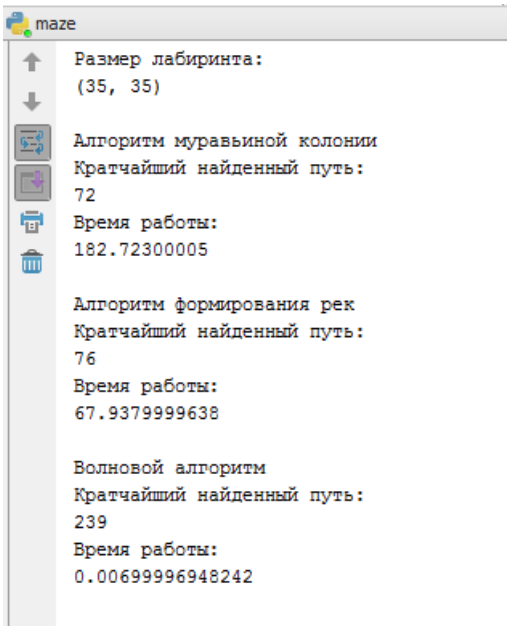

Рисунок 2 – Результат работы программы

Рисунок 3 – Кратчайший путь, найденный Муравьиным алгоритмом

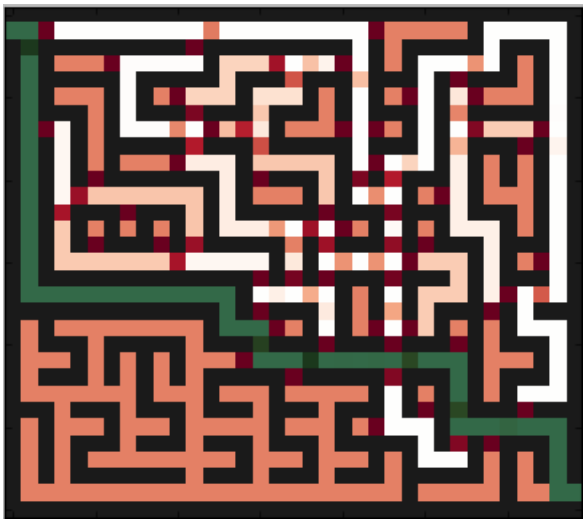

Рисунок 4 – Кратчайший путь, найденный Алгоритмом формирования рек

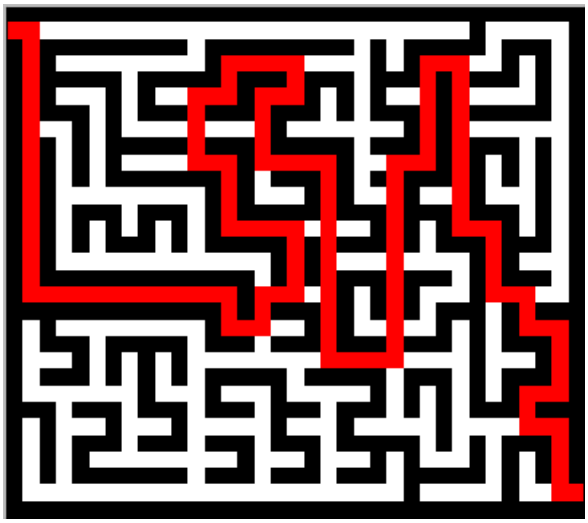

Рисунок 5 – Кратчайший путь, найденный Волновым алгоритмом

3.2 Сравнение алгоритмов

Все алгоритмы, рассмотренные выше, были сравнены по времени работы и длине находимого ими кратчайшего пути в лабиринте. Ниже представлены результаты для одиннадцати различных размеров лабиринтов. 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 400 клеток и более, АФР

затрачивает времени в два раза меньше, чем АМК. Для лабиринтов площадью более 1000 клеток, скорость их работы начинается отличаться приблизительно в 3 раза, из чего можно сделать вывод, что АФР является более эффективным роевым алгоритмом для поиска кратчайшего пути. В сравнении с алгоритмам роевого интеллекта, Волновой алгоритм показывает не самые лучшие результаты по длине находимого кратчайшего пути(находя пути в среднем в $2,5$ раза более длинные, чем другие алгоритмы). Но его преимуществом является быстрота работы даже в лабиринтах больших размеров, поскольку затрачиваемое время для поиска кратчайшего пути составляет менее секунды, даже в лабиринтах площадью более 1500 клеток.

ЗАКЛЮЧЕНИЕ

В данной работе был создан алгоритм генерации лабиринтов, реализованы такие алгоритмы роевого интеллекта, как Алгоритм муравьиной колонии и Алгоритм формирования реки для поиска кратчайшего пути в лабиринте, создан графический интерфейс для каждого из этапов. Для сравнения эффективности алгоритмов был также реализован Волновой алгоритм — наиболее популярный алгоритм поиска путей в лабиринте, не относящийся к роевому интеллекту. Сравнительный анализ алгоритмов по времени работы и длине находимого кратчайшего пути показал, что роевые алгоритмы могут успешно применяться для поиска кратчайших путей в лабиринтах небольших размеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Керн, Г.* Лабиринт: основные принципы, гипотезы, интерпретации / Г. Керн. — Азбука-классика, 2007.

2 *Papert, S.* Uses of technology to enhance education [электронный ресурс]. — 1973. — URL: ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-298.pdf (дата обращения: 05.05.2016).

3 *van Vliet, D.* Improved shortest path algorithms for transport networks [электронный ресурс]. — 1977. — URL: http://www.sciencedirect.com/science/article/pii/0041164778901028 (дата обращения: 05.05.2016).

4 *Мозговой, М.* Занимательное программирование: Самоучитель / М. Мозговой. — Питер, 2004.

5 *Garnier, S.* The biological principles of swarm intellgience / S. Garnier. — 2007.

6 *Dorigo, M.* Ant colony optimization. — 2004.

7 *Rabanal, P.* Using river formation dynamics to design heuristic algorithms [электронный ресурс]. — 2007. — URL: https://svn-d1.mpi-inf.mpg.de/AG1/MultiCoreLab/papers/RabanalRodriguezRubio07%20-%20River%20Formation%20Dynamics.pdf (дата обращения: 16.05.2016).