Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

**Факультет Информатика и системы Управления (ИУ)**

**Кафедра «Информационная безопасность» (ИУ8)**

**Отчет по научно-исследовательской работе студента**

на тему Сравнительный анализ эффективности решения задачи коммивояжера генетическим и муравьиным алгоритмом.

ФИО студента: Белоголов Александр Николаевич

Группа: ИУ8-33 Курс:  *2*

Специальность:  *10.05.03.01 «Информационная безопасность автоматизированных систем»*

Научный руководитель НИРС Колесников А.В.

|  |  |  |  |
| --- | --- | --- | --- |
| Работа выполнена |  |  | А.Н. Белоголов |
|  | дата | подпись студента | И. О. Фамилия студента |
| Допуск к защите |  |  | А.В. Колесников |
|  | дата | подпись научного руководителя | И. О. Фамилия научного руководителя |

Дата защиты НИРС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Результаты защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва

2023

**РЕФЕРАТ**

Отчёт 19 с., 1 рис., 4 ч., 5 источников.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА ГЕНЕТИЧЕСКИМ И МУРАВЬИНЫМ АЛГОРИТМОМ.

Цель работы — реализовать генетический и муравьиный алгоритмы решения задачи коммивояжера и сравнить их сильные и слабые стороны.

В процессе работы были изучены генетический и муравьиный алгоритмы, а затем и реализованы на языке C#. Были проведены испытания реализаций алгоритмов.

В результате были исследованы сильные и слабые стороны реализаций генетического и муравьиного алгоритмов. На основе этих данных было проведено сравнение эффективности решения этими алгоритмами задачи коммивояжера.

4

**СОДЕРЖАНИЕ**

[ОПРЕДЕЛЕНИЯ 6](#_Toc154599555)

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 8](#_Toc154599556)

[ВВЕДЕНИЕ 9](#_Toc154599557)

[ОСНОВНАЯ ЧАСТЬ 12](#_Toc154599558)

[1 Этапы выполнения работы 12](#_Toc154599559)  
[2 Реализация генетического алгоритма 13](#_Toc154599559)

[2.1 Реализация вспомогательного класса пути (Path) 14](#_Toc154599559)

[2.2 Реализация метода создания популяции 14](#_Toc154599560)

[2.3 Реализация алгоритма скрещивания 15](#_Toc154599561)

[2.4 Реализация алгоритма мутации 15](#_Toc154599562)

[2.5 Интеграция дочерних хромосом, и реализация естественного отбора в популяции 15](#_Toc154599562)

[3 Реализация муравьиного алгоритма 16](#_Toc154599563)

[3.1 Реализация метода выбора пути муравьем. 17](#_Toc154599564)

[3.2 Реализация метода, отвечающего за расчет отложений феромона после итерации. 17](#_Toc154599565)

[3.3 Реализация функции, возвращающей самый эффективный из найденных путей 18](#_Toc154599565)

[4 Сравнение эффективности реализаций генетического и муравьиного алгоритмов 19](#_Toc154599559)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc154599566)

# ОПРЕДЕЛЕНИЯ

**Общие понятия:**

* **Задача коммивояжера (почтальона)** – одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске самого выгодного маршрута, проходящего через указанные города (вершины графа) хотя бы по одному разу с последующим возвратом в исходный город.
* **Матрица смежности** – матрица, где значение ячейки с индексами i и j соответствует длине пути между вершинами i и j графа. На главной диагонали этой матрицы бесконечности (или нули, так как не имеет смысла идти из вершины в нее же).
* **Длинна маршрута** – сумма длин последовательных ребер между вершинами, которые содержит маршрут.
* **Классом NP** (от англ. non-deterministic polynomial) называют множество задач разрешимости, решение которых возможно проверить на машине Тьюринга за время, не превосходящее значения некоторого многочлена от размера входных данных, при наличии некоторых дополнительных сведений (так называемого сертификата решения).

**Термины, используемые в генетическом алгоритме:**

* **Генетический алгоритм** (англ. genetic algorithm) — эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе.
* **Хромосома** – последовательность вершин, которые образуют маршрут.
* **Ген –** ячейка в хромосоме.
* **Популяция** – множество хромосом, то есть маршрутов.
* **Особь –** набор хромосом, которые удовлетворяют решению.
* **Скрещивание –** операция, при которой две хромосомы обмениваются своими частями.
* **Мутация –** случайное изменение одной или нескольких позиций в хромосоме.
* **Приспособленность –** длинна маршрута.

**Термины, используемые в муравьином алгоритме:**

* **Муравьиный алгоритм** – это эвристический оптимизационный метод, инспирированный поведением муравьев при поиске оптимальных путей к источнику пищи. Алгоритм моделирует взаимодействие муравьев, используя следы феромонов, чтобы эффективно искать приближенные решения для задач комбинаторной оптимизации.
* **Муравей** – последовательность вершин, которые образуют маршрут.
* **Колония** – множество муравьев, то есть маршрутов.
* **Феромон –** численная характеристика ребра графа (помимо веса), влияющая на вероятность выбора пути муравьем.
* **Близость** – величина, обратная к дальности.

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **GA (англ.)** | – | (Genetic Algorithm) сокращение для генетического алгоритма |
| **ACO (англ.)** | – | (Ant Colony Optimization) сокращение для муравьиного алгоритма. |
| **NP (англ.)** | – | (Non-deterministic Polynomial) класс алгоритмов, которые можно решить за полиномиальное время. |
|  |  |

# ВВЕДЕНИЕ

Задача коммивояжёра – одна из интереснейших подзадач комбинаторной оптимизации. Необходимость ее решения встречается, к примеру, в логистике, когда транспортной компании, доставляющей товары в некоторые пункты, необходимо выстроить наиболее эффективный алгоритм доставки по ним.

Коммивояжёр (фр. commis voyageur) — бродячий торговец. Коммивояжёру, чтобы распродать нужные и не очень нужные в хозяйстве товары, следует объехать n пунктов и в конце концов вернуться в исходный пункт. Требуется определить наиболее выгодный маршрут объезда. В качестве меры выгодности маршрута (точнее говоря, невыгодности) может служить суммарное время в пути, суммарная стоимость дороги, или, в простейшем случае, длина маршрута.

Совершенно очевидно, что задача может быть решена перебором всех вариантов объезда и выбором оптимального. Беда в том, что количество возможных маршрутов очень быстро возрастает с ростом n (оно равно n! – количеству способов упорядочения пунктов). К примеру, для 100 пунктов количество вариантов будет представляться 158-значным числом — не выдержит ни один калькулятор! Мощная ЭВМ, способная перебирать миллион вариантов в секунду, будет биться с задачей на протяжении примерно 3\*10144 лет. Увеличение производительности ЭВМ в 1000 раз даст хоть и меньшее в 1000 раз, но по-прежнему чудовищное время перебора вариантов. Не спасает ситуацию даже то, что для каждого варианта маршрута имеется 2n равноценных, отличающихся выбором начального пункта (n вариантов) и направлением обхода (2 варианта). Перебор с учётом этого наблюдения сокращается незначительно — до вариантов.

Может быть, алгоритм, основанный на полном переборе вариантов, не является самым эффективным (в смысле быстродействия) для решения задачи коммивояжёра? Увы, доказано, что не существует алгоритма решения, имеющего степенную сложность (то есть требующего порядка операций для некоторого ) — любой алгоритм будет хуже. Всё это делает задачу коммивояжёра безнадёжной для ЭВМ с последовательным выполнением операций, если хоть сколько-нибудь велико.

В таком случае следует отказаться от попыток отыскать точное решение задачи коммивояжёра и сосредоточиться на поиске приближённого — пускай не оптимального, но хотя бы близкого к нему. В виду большой практической важности задачи полезными будут и приближённые решения.

Заметим, что интеллект человека, не вооружённый вычислительной техникой, способен отыскивать такие приближённые решения задач, требующих огромного перебора вариантов в поисках оптимального. Вспомним хотя бы шахматы. Человек может весьма успешно соперничать в этой игре с вычислительной машиной либо вовсе не прибегая к перебору, либо сводя его к минимуму. Человек руководствуется при этом интуицией и набором *эвристик* (находок) — правил, которые обычно помогают в решении задач, хотя эффективность таких правил и не имеет достаточного обоснования. В качестве подобной универсальной эвристики можно упомянуть категорический императив Канта: «поступай с другими так, как тебе хотелось бы, чтобы поступали с тобой». Другой, более приземлённый пример даёт золотое правило валютного спекулянта: «когда все продают доллары, ты покупай, а когда все покупают — продавай».

Многие природные процессы решают задачи выбора оптимального варианта из огромного (даже, возможно, бесконечного) множества вариантов. Например, тяжёлая гибкая однородная цепочка, подвешенная за концы на двух гвоздиках, из всевозможных доступных форм принимает именно ту, которая соответствует минимуму потенциальной энергии силы тяжести (которая пропорциональна высоте центра тяжести цепочки). Причём цепочке для поиска нужной формы (она называется катеноидой, или цепной линией) требуется времени гораздо меньше, чем человеку, составляющему и решающему дифференциальное уравнение Эйлера — Лагранжа для нахождения этой самой катеноиды. Мыльная плёнка, натянутая на проволочный контур, принимает форму, соответствующую минимуму внутренней энергии плёнки (состоящей, в основном, из потенциальной энергии сил поверхностного натяжения, пропорциональной площади плёнки). Световой луч в прозрачной (возможно, неоднородной) среде, преломляясь, отыскивает кратчайший путь (требующий наименьшего времени прохождения любого своего участка) с учётом скорости света в каждой точке среды, через которую он проходит. Вещество, кристаллизуясь из расплава, постепенно принимает ту кристаллическую форму, которая минимизирует опять же внутреннюю энергию, складывающуюся из энергий попарного взаимодействия молекул. В последнем примере молекулы вещества, совершающие хаотическое тепловое движение, которое замедляется по мере остывания, постепенно «нащупывают» нужную, минимальную в энергетическом смысле конфигурацию среди огромного количества вариантов расположения молекул. Биологическая эволюция совершенствует виды, снижая вероятность выживания (и, соответственно, передачи потомству генетической информации) менее приспособленных особей.

Все эти соображения ведут нас к эвристике: «хочешь приближённо решить задачу — смоделируй (например, с помощью ЭВМ) природный процесс, решающий подобную задачу». Вот несколько конкретных приложений этой эвристики: «нужен эллипс — посвети фонариком на пол, слегка наклонив его (фонарик или пол)». Или «налей в цилиндрический или конический стакан воды и чуть наклони». Или «возьми батон колбасы и разрежь наискосок». «Нужна синусоида — заверни колбасу в бумагу и разрежь вместе с бумагой, а затем разверни бумажный лист». Или «присоедини колебательный контур к осциллографу» — колебательный контур мгновенно решает дифференциальное уравнение колебаний, а решения этого уравнения — синусы и косинусы. «Хочешь вычислить определённый интеграл — вырежи криволинейную трапецию из бумаги, взвесь её и подели на массу единичного бумажного квадратика». Описанные выше наблюдения позволяют считать природные процессы вычислительными машинами (такие машины называют аналоговыми), вполне пригодными для решения многих важных задач. Аналоговые вычислительные машины можно использовать непосредственно, а можно принципы их работы положить в основу весьма эффективных алгоритмов для традиционных, цифровых ЭВМ. Единственное, что может пострадать при таком моделировании — точность решения задачи.

# ОСНОВНАЯ ЧАСТЬ

**1 Этапы выполнения работы:**

1. Изучение постановки задачи коммивояжера и некоторых переборных вариантов ее решения.
2. Изучение механизма работы генетического алгоритма решения задачи коммивояжера.
3. Реализация генетического алгоритма на языке C#.
4. Изучение механизма работы муравьиного алгоритма решения задачи коммивояжера.
5. Реализация муравьиного алгоритма на языке C#.
6. Исследование сильных и слабых мест обоих алгоритмов.
7. Сравнение эффективности алгоритмов и подведение итогов.

**2 Реализация генетического алгоритма**

Как говорилось в ведении, в среднем наиболее эффективными алгоритмами поиска маршрута являются эвристические алгоритмы, и генетический алгоритм – один из них.

Алгоритм основан на принципе естественного отбора генов в природе.

Алгоритм работает следующим образом: сначала выбираются начальные параметры алгоритма, а именно: матрица смежности, на которой мы будем искать кратчайший маршрут, размер популяции, минимальное количество поколений, через которое мы наиболее вероятно будем знать ответ, процент мутаций (в моей реализации также сразу было указано количество вершин в графе, но, вообще говоря, это можно понять просто из матрицы смежности). Также стоит отметить, что эффективность алгоритма сильно зависит от успешного выбора параметров размера популяции, числа поколений и процента мутаций. Затем происходит генерация популяции нулевого поколения – она происходит посредством выбора общей начальной вершины для каждой хромосомы, и затем заполнения маршрута случайным образом. После этого происходит процесс скрещивания двух случайных хромосом из популяции с получением двух их дочерних хромосом, содержащих хромосомы обоих родителей. Затем процесс случайной мутации каждой из дочерних хромосом, шанс которой определяется процентом мутаций. И в конце концов отбор среди всех хромосом в популяции с учетом новых двух – дочерних. Наименее приспособленные две хромосомы из популяции удаляются. Процесс повторяется заданное число поколений.

## 2.1 Реализация вспомогательного класса пути (Path)

Прежде чем писать класс для реализации самого генетического алгоритма, полезно будет реализовать интерфейс класса пути коммивояжера. Это нужно для того, чтобы потом можно было легко обращаться к полю класса, содержащему длину пути, а также можно добавить реализацию интерфейса IComparable, чтобы по длине пути можно было еще и легко сортировать наши хромосомы.

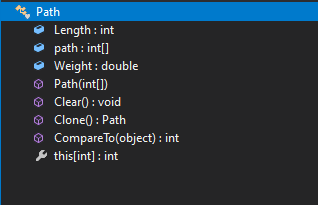


Рисунок 1. Представление шаблона для класса Path

## 2.2 Реализация метода создания популяции

Перейдем к созданию самого класса для реализации генетического алгоритма. Я назвал его Genetic. Разберем реализации основных его методов. Первым будет метод создания популяции (GeneratePop()).

Выбираем общую начальную вершину – пусть это будет вершина с индексом 0. Тасуем все оставшиеся вершины алгоритмом Фишера – Йетса, и заполняем нашу первую хромосому, получая случайный маршрут следования коммивояжера. Далее создаем по этому же алгоритму хромосомы до тех пор, пока количество хромосом в нашей популяции не будет равно заданному размеру популяции. В конце сортируем наш массив хромосом по параметру их приспособленности.

## 2.3 Реализация алгоритма скрещивания

Выбираем две случайные хромосомы из популяции – они будут родительскими хромосомами. Выбираем границу скрещивания (в моем случае это половина хромосомы с округлением вниз). Создаем две дочерние хромосомы. Гены до границы скрещивания первой дочерней хромосомы заполняем генами до границы скрещивания первого родителя. Для второй – второго. Далее первую дочернюю хромосому после границы скрещивания заполняем подряд генами второй родительской хромосомы, не имеющимися в дочерней, и остаток генов добиваем недостающими генами из первой родительской хромосомы. Порядок следования генов всюду сохраняется! То же самое повторяем для второй дочерней хромосомы, но наоборот – сначала берем гены первого родителя, потом второго.

## 2.4 Реализация алгоритма мутации

Для каждого дочернего гена может произойти случайная мутация. Генерируем случайное число от 0 до 100. Если оно меньше заданного процента мутации – активируем алгоритм мутации. Он меняет два случайных гена в хромосоме местами (начальную вершину не трогаем).

**2.5 Интеграция дочерних хромосом, и реализация естественного отбора в популяции**

Добавляем наши новые дочерние хромосомы в популяцию, сортируем ее по приспособленности, и выкидываем две наименее приспособленные особи. Повторяем весь этот алгоритм заданное число поколений. В результате наиболее приспособленная особь покажет наиболее эффективный маршрут.

## 3 Реализация муравьиного алгоритма

Муравьиный алгоритм – другой эвристический алгоритм для решения задачи коммивояжера. Его механизм работы основан на выводах из наблюдений за муравьями: они всегда достаточно быстро выбирают наиболее эффективные маршруты в своих колониях, благодаря отложениям из феромона, которые они оставляют за собой после прохода.

Алгоритм работает следующим образом: задаются параметры алгоритма, а именно неотрицательные коэффициенты α и β (в моем случае 1 и 4, но, вообще говоря, их можно подбирать и под каждый граф отдельно опытным путем), от которых будет зависеть то, насколько важными для муравья при выборе маршрута будут отложения феромона и длины путей соответственно, коэффициент испарения феромона ρ (должен быть от 0 до 1, в моем случае равный 0.2), положительный коэффициент отложения феромона муравьем Q, количество муравьев в колонии (рекомендуется брать равным количеству городов), количество итераций алгоритма. Заметим, что от выбора параметров, как и в случае с GA, зависит скорость работы алгоритма и его точность. Затем запускается первая итерация алгоритма: каждый муравей в колонии строит свой маршрут. Для стабильности работы алгоритма также лучше запускать каждого следующего муравья из новой вершины, чтобы избежать возникновения систематически повторяющихся неэффективных маршрутов. После того как все муравьи проложат свой маршрут, рассчитывается испарение феромона, и наложение его от прохода муравьев. После этого, на основе отложений феромона с прошлой итерации, аналогично запускается следующая. Таким образом, количество феромона на самом выгодном пути будет систематически возрастать, и в конце концов по отложениям феромона мы сможем понять самый эффективный маршрут.

## 3.1 Реализация метода выбора пути муравьем.

Этот метод будет принимать на вход текущую вершину, в которой находится муравей, а также массив вершин, которые он уже посетил. Все, что мы будем делать в этом методе, это смотреть на характеристики путей во все вершины, которые мы еще не посетили, и опираясь на это, случайным образом выбирать из них следующую вершину, в которую последует муравей.

Вероятность выбора пути из i-ой вершины в j-ую муравьем равна:

Где – количество феромона на грани ij, возведенная в степень α;  
 – близость между вершинами i и j, возведенная в степень β;

m – все доступные (еще не посещенные) вершины.

## 3.2 Реализация метода, отвечающего за расчет отложений феромона после итерации.

После того, как все муравьи на итерации проложили свои маршруты, для всех граней производится перерасчет феромона по формуле:

Где t – номер итерации;

– количество отложенного феромона k-ым муравьем (равно 0, если муравей не проходил между вершинами i и j).

**3.3 Реализация функции, возвращающей самый эффективный из найденных путей.**

Повторяем итерации заданное количество раз, и затем, проходим по пути, содержащему наибольшее количество феромона. Этот путь и будет ответом.

**4 Сравнение эффективности реализаций генетического и муравьиного алгоритмов.**

Так как оба алгоритма принадлежат классу сложности NP и являются эвристическими, численно сравнить их сложности не получится, так что будем сравнивать на практике. Надёжность алгоритмов была рассчитана путём усреднения результатов после 30 запусков каждого из них на выборках из случайных матриц размером 30x30, обоим алгоритмам давалось 30 хромосом (муравьев) и 10000 итераций.

Средние длинны маршрутов оказались примерно одинаковыми, муравьиный алгоритм чаще показывал наиболее короткие маршруты в сравнении с генетическим, но тратил намного больше времени. Также было выяснено, что генетический алгоритм менее чувствителен к подбору параметров, нежели муравьиный, однако это может быть как плюсом, так и минусом, ведь успешный подбор параметров в муравьином алгоритме выдавал более точные значения.

**ЗАКЛЮЧЕНИЕ**

По итогам работы был изучен механизм работы генетического и муравьиного алгоритма. По результатам их сравнения можно сказать, что оба алгоритма достаточно быстры и точны в случае решения задачи коммивояжера. Если решение задачи сильно ограничено по времени – лучше выбирать генетический алгоритм, ведь он более гибкий, быстрый, и менее чувствителен к неудачному подбору параметров. В ином случае – муравьиный алгоритм покажет себя лучше, чем генетический.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Параллельные генетический и муравьиный алгоритмы для решения задачи коммивояжера [Электронный ресурс]. – Режим доступа: URL: <https://cyberleninka.ru/article/n/parallelnye-geneticheskiy-i-muravinyy-algoritmy-dlya-resheniya-zadachi-kommivoyazhera> (дата обращения: 27.12.2023)
2. Исследование эффективности параллельного муравьиного алгоритма на задаче коммивояжера [Электронный ресурс]. – Режим доступа: URL: <https://cyberleninka.ru/article/n/issledovanie-effektivnosti-parallelnogo-muravinogo-algoritma-na-zadache-kommivoyazhera> (дата обращения: 27.12.2023)
3. Задача коммивояжера и генетические алгоритмы [Электронный ресурс]. – Режим доступа: URL: <https://cyberleninka.ru/article/n/zadacha-kommivoyazhera-i-geneticheskie-algoritmy> (дата обращения: 27.12.2023)
4. Сравнение эффективности генетического и муравьиного алгоритмов при решении задач комбинаторной оптимизации [Электронный ресурс]. – Режим доступа: URL: <https://masters.donntu.ru/2018/fknt/savkin/library/article_translate.htm> (дата обращения: 27.12.2023)
5. О классах сложности алгоритмов [Электронный ресурс]. – Режим доступа: URL: <https://cyberleninka.ru/article/n/o-klassah-slozhnosti-algoritmov> (дата обращения: 27.12.2023)