Санкт-Петербургский Политехнический Университет
Высшая школа прикладной математики и вычислительной физики, ФизМех
01.03.02 Прикладная математика и информатика

Курсовая работа
тема «Муравьиный алгоритм: вариация элитарной муравьиной системы и Max-Min муравьиной системы»
дисциплина ”Методы оптимизации”

Выполнили студенты гр. 5030102/20401 Тыщенко А. Д.
Конькова А. И.

Родионова Е. А.

Преподаватель

Санкт-Петербург
2025

Содержание

1 Введение

Задача коммивояжера (TSP, Traveling Salesman Problem) является одной из классических задач комбинаторной оптимизации, которая заключается в нахождении кратчайшего маршрута, проходящего через заданный набор городов и возвращающегося в исходный город. Эта задача имеет широкое применение в различных областях, таких как логистика, планирование маршрутов, проектирование интегрированных систем и многие другие. Однако, несмотря на свою простоту в формулировке, задача коммивояжера относится к NP-трудным задачам, что делает ее решение сложным для больших наборов данных.[3]

В последние десятилетия для решения задач комбинаторной оптимизации, включая задачу коммивояжера, активно применяются метаэвристические методы. Одним из таких методов является муравьиный алгоритм, который имитирует поведение муравьев при поиске пищи. Этот алгоритм основывается на принципах коллективного поведения и позволяет находить приближенные решения для сложных задач оптимизации. В рамках муравьиного алгоритма выделяются различные подходы, среди которых особое внимание уделяется элитарной системе и max-min муравьиному алгоритму.[3]

Элитарная система предполагает, что лучшие найденные решения сохраняются и используются в последующих итерациях, что позволяет ускорить процесс сходимости и улучшить качество решений. Max-min муравьиный алгоритм, в свою очередь, вводит механизм, который ограничивает максимальную и минимальную длину пути, что способствует более равномерному распределению феромонов и улучшает исследование пространства решений.

Целью данной курсовой работы является сравнение эффективности элитарной системы и max-min муравьиного алгоритма при решении задачи коммивояжера. В ходе работы будут рассмотрены основные принципы работы каждого из алгоритмов, проведены эксперименты на различных наборах данных и проанализированы полученные результаты. Ожидается, что результаты исследования позволят выявить преимущества и недостатки каждого из подходов, а также определить условия, при которых один из алгоритмов может быть предпочтительнее другого.

2 Формулировка задачи и её формализация

2.1 Формулировка задания

Для заданных N городов продавец, отправляющийся из своего родного города, должен посетить каждый город ровно по одному разу, а затем вернуться домой. Требуется найти порядок тура таким образом, чтобы общее пройденное расстояние было минимальным. Для представления TSP будем использовать полный взвешенный граф $G = (N, E)$, где N - множество из n городов, а E - множество ребер (путей), полностью соединяющих все города. Каждому ребру $(i, j) \in E$ присваивается стоимость d_{ij}, которая представляет собой расстояние между городами i и j. Повторное посещение городов запрещено.

2.2 Формализация задания

$n \geq 0$ — количество городов
$M_{n,n}$ — матрица времени, $m_{ij} \geq 0, \quad i, j = \overline{1, n}$
$S \geq 0$ — ограничение по времени
$C = ||c_{ij}||$ — матрица расстояний между городами, $c_{i,j} \geq 0, \quad i, j = \overline{1, n}$

$$\begin{cases} \sum_{i=1}^{k} c_i \to min; \quad k = \overline{1, n} \\ \sum_{i=1,j=1}^{k} m_{ij} \leq S \end{cases}$$

3 Решение задачи

3.1 Общие сведения

Первоначально идею муравьиного алгоритма предложил Марко Дориго в 1992 году в его докторской диссертации, первый алгоритм был направлен на поиск оптимального пути в графе, основанном на поведении муравьев, ищущих путь между своей колонией и источником пищи для решения задач оптимизации. В естественном мире муравьи бродят хаотично и, найдя пищу, возвращаются в свою колонию, прокладывая феромонные тропы. Другие же муравьи, находя тропы с феромонами, используют эту информацию, тем самым укрепляя феромонную тропу.
1. Чем короче путь до источника пищи, тем меньше времени понадобится на пермещение муравьям — следовательно, тем быстрее оставленные на нем следы будут заметны.
2. В итоге, чем больше муравьев проходит по определенному пути, и чем этот путь короче, тем он становится более привлекательным для остальных муравьев.

3.2 Описание

Каждый муравей в данной задаче рассматривается как отдельный (и независимый от других) коммивояжёр, решающий проблему, проходя в течение одной итерации весь маршрут.
”Случайность” реализуется с помощью вероятностного правила, с помощью которого обеспечивается положительная обратная связь: вероятность того, что ребро графа включено в маршрут муравья, пропорциональна значению его феромона.Для имитации поведения муравьёв объем феромонов, размещённых на ребре графа, принимается обратно пропорциональным длине пути. Однако положительная обратная связь приводит к застою: в этом случае все муравьи выбирают один неоптимальный путь. Чтобы избежать этого, существует отрицательная обратная связь: через ферромон вводится испарение. Интенсивность испарения не должна быть слишком высокой; в противном случае область поиска сузится (ситуации, когда колония преждевременно “забывает” свой опыт, полученный в прошлом (потеря памяти)).[3]

Для каждого муравья проход из города i в город j зависит от следующих трех компонентов: список запретов (tabu list) или память муравья, видимость и следы виртуальных феромонов.

Список запретов - это структура данных, которая сохраняет список уже посещенных городов, которые не следует посещать снова. Этот список увеличивается в размерах на каждом шаге и устанавливается равным нулю в начале каждой итерации алгоритма. Обозначение: $J_{i,k}$ - список городов, которые еще предстоит посетить k-ому муравью, расположенному в i-ом городе.

Видимость - является обратной величиной к расстоянию:

$$\eta_{i,j} = \frac{1}{D_{i,j}},$$

где $D_{i,j}$ - это расстояние между городами i и j. Видимость - это локальное статическое значение, отражающее эвристическое желание переехать из города i в город j: чем ближе город, тем сильнее желание его посетить.

Муравьиный алгоритм можно классифицировать как вероятностный, то есть он дает только приближенное решение, не гарантируя его оптимальности.

3.3 Формализация алгоритма для общего случая

1. Создаем муравьёв и задаём начальные значения феромонов.
m — количество муравьёв
n — количество городов
$C_{n,n}$ — матрица расстояний
Инициализация параметров:

- α, β —константные параметры, которые описывают вес феромонного следа и посещаемость при выборе маршрута
- $Q > 0$ — параметр, имеющий значение порядка цены оптимального решения
- $p \in [0, 1]$ — коэффициент испарения
- τ_0 — начальный уровень феромона

Инициализация видимости $\eta_{i,j}$ и уровень феромона $\tau_{i,j}(t)$
$\eta_{i,j} = \frac{1}{C_{i,j}}, C_{i,j}, i \neq j$
$\tau_{i,j}(0) = \begin{cases} \tau_0, i \neq j \\ 0, i = j \end{cases}$

2. Вероятность перемещения k-го муравья на итерации t из города i в город j вычисляется по следующему правилу:

$$P_{i,j,k}(t) = \begin{cases} \frac{(\tau_{i,j}(t))^{\alpha}(\eta_{i,j})^{\beta}}{\sum_{l \in J_{i,k}}} \\ 0, j \notin J_{i,k} \end{cases}$$

Когда $\alpha = 0$, выбирается ближайший город, что соответствует жадному алгоритму в классической теории оптимизации. Когда $\beta = 0$, учитывается только след феромона, что означает, что все муравьи выбирают один неоптимальный маршрут. Чтобы обеспечить хорошую динамику оптимизации, рекомендуется устанавливать $\alpha \geq \beta$.

Стоит отметить, что $P_{i,j,k}(t)$ определяет вероятности выбора конкретного города. Сам выбор осуществляется по принципу "колеса рулетки": каждый город на нём имеет свой собственный сектор с площадью, соответствующий вероятности $P_{i,j,k}(t)$.

3. Феромон, откладываемый k-ым муравьём, использующим ребро (i, j)

$$\Delta\tau_{i,j,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, (i,j) \in T_k(t) \\ 0, (i,j) \notin T_k(t) \end{cases}$$ где $T_k(t)$ — это маршрут муравья k на итерации, $L_k(t)$ — длина маршрута.

4. Обновляем уровень феромона в соответствии с формулой:

$$\tau_{i,j}(t+1) = (1-p)\tau_{i,j}(t) + \sum_{k=1}^{m} \Delta\tau_{i,j,k}(t)$$

3.4 Элитарная муравьиная система

3.4.1 Отличия элитарной модификации от общего случая

Одной из ключевых проблем, с которой сталкивается муравьиный алгоритм, является положительная обратная связь. В условиях положительной обратной связи все муравьи могут сосредоточиться на одном неоптимальном пути, что приводит к застою и снижению качества решений. Чтобы избежать этой проблемы, была разработана элитарная модификация муравьиного алгоритма.

Элитарная модификация направлена на улучшение качества решений и ускорение сходимости алгоритма. Основная идея заключается в том, чтобы сохранить лучшие найденные решения (или пути) и обеспечить их более высокую вероятность выбора в последующих итерациях. Это позволяет алгоритму использовать уже найденные хорошие решения, что значительно повышает вероятность нахождения глобального оптимума.[2]

Причины применения элитарной модификации разнообразны. Во-первых, она позволяет улучшить качество решений, так как лучшие пути сохраняются и становятся более предпочтительными. Во-вторых, это ускоряет сходимость алгоритма, так как муравьи могут опираться на уже найденные эффективные маршруты. В-третьих, элитарная модификация помогает избежать ситуации, когда алгоритм застревает в локальных минимумах, так как лучшие решения продолжают оказывать влияние на выбор путей.[2]

Элитарная модификация отличается от общего случая муравьиного алгоритма несколькими ключевыми аспектами. Во-первых, в общем случае алгоритм не сохраняет лучшие решения, а просто обновляет феромоны на основе всех найденных путей. В элитарной модификации лучшие решения сохраняются и получают дополнительное внимание. Во-вторых, вероятность выбора пути в общем случае зависит от феромонов и эвристической информации, тогда как в элитарной модификации вероятность выбора лучших путей увеличивается. Наконец, обновление феромонов в элитарной модификации может происходить более агрессивно для лучших путей, что позволяет им сохранять свое влияние на выбор путей.

3.4.2 Формализация элитарной модификации

Правило обновления феромонов принимает вид:

$\tau_{i,j}(t+1) = (1-p)\tau_{i,j}(t) + p\Delta\tau_{i,j,e}(t)$

где (i,j) — край наилучшего маршрута (либо на текущей итерации, либо с начала алгоритма)

$\Delta\tau_{i,j,e}(t) = \begin{cases} \frac{Q}{L^+}, (i,j) \in T^+ \\ 0, (i,j) \notin T^+ \end{cases}$

"Элитные" муравьи выделяют феромоны только по рёбрам лучшего найденного маршрута T^+. Для поставленной задачи берётся значение феромона "элитного" муравья на каждом ребре маршрута T^+ равным $\frac{Q}{L^+}$, где L^+ — длина маршрута T^+.

Шаг модифицируется следующим образом: k-й муравей перемещается с вероятностью q_0 из города i в наиболее оптимальный город $z \in J_{i,k}$ и с вероятностью $(1-q_0)$ выбирает город j по шагу $P_{i,j,k}(t)$. Чем больше q_0, тем выше эффективность использования опыта, полученного колонией муравьёв при синтезе новых маршрутов. Наиболее привлекательный город определяется так:

$z = \arg\max_{j \in J_{i,k}} (\tau_{i,j}(t))^\alpha (\eta_{i,j})^\beta$

На каждой итерации при переходе из города i в город j муравей "съедает" какое-то количество феромонов с этого ребра графа. Это ребро теряет свою привлекательность для других муравьёв: они начинают рассматривать другие маршруты. Решения становятся более разнообразными благодаря динамичному обновлению распределения феромонов.
Также:

1. Уровень феромонов на рёбрах обновляется не только на каждой итерации, но и при переходе муравьёв из узла в узел.
2. В конце итерации уровень феромонов повышается только на лучшем из найденных путей.
3. Муравей либо с определённой вероятностью выбирает лучшее ребро, либо производит выбор как в классическом алгоритме.

3.4.3 Тестовый пример для элитарной муравьиной системы

Матрица времени:

$$\begin{bmatrix} 14 & 0 & 19 & 26 & 30 \\ 0 & 14 & 36 & 34 & 22 \\ 36 & 19 & 0 & 15 & 36 \\ 34 & 26 & 15 & 0 & 21 \\ 22 & 30 & 36 & 21 & 0 \end{bmatrix}$$

- Количество поколений: 1
- Количество муравьёв в поколении: 2
- Коэффициент запаха $\alpha = 0.1$
- Коэффициент расстояния $\beta = 2$
- Глобальный коэффициент обновления $e = 0.1$
- Локальный коэффициент обновления $p = 0.1$
- Количество выпускаемых феромонов $Q = 1$
- Коэффициент баланса перехода между городами $q = 0.9$

1. Для начала проинициализируем каждый элемент матрицы феромонов начальным значением $\tau_{i,j}(0) = \phi = \frac{n \cdot Q}{2} = 10$, а затем обнулим диагональные элементы.
2. Затем для каждого из муравьёв подсчитаем вероятности перехода в k-тый город из первого города по формуле из алгоритма выше. Для тех городов, где муравей уже был, вероятность считаем равной 0, также равной 0 будет вероятность перехода в k-тый город, если расстояние до него больше, чем оставшееся время муравья.
3. Пример расчета вероятности перехода первого муравья из первого города во второй:

$$P_{10,1} = \frac{(\tau_{1,2}(t))^{\alpha} \cdot (\eta_{1,2})^{\beta}}{\sum_{l \in J_{i,k}} (\tau^{\alpha} + \eta^{\beta})} = \frac{\left(\frac{1}{10}\right)^1 \cdot \left(\frac{1}{14}\right)^2}{30} = 2.38095 \times 10^{-4}$$

4. Таблица вероятностей переходов для первого муравья:

№ вершины	1	2	3	4	5
1	0	2.38095e-4	4.62963e-5	5.76701e-6	6.88705e-5
2	0	0	1.66205e-4	9.86193e-6	3.7037e-5
3	0	0	0	2.96296e-5	2.64550e-4
4	0	0	0	0	0

5. Таблица сгенерированных $rand_1$:

1	2	3
0.568823660872193	0.469390641058206	0.337122644398882

6. На каждой итерации сгенерируем случайное число $rand_k$, лежащее от 0 до 1. Если все вероятности перехода для данного муравья равны 0, значит, муравей закончил свой путь, и мы переходим к следующему. Иначе, если случайная величина, соответствующая муравью, меньше или равна коэффициенту $q = 0.9$, то муравей переходит в вершину с наибольшей вероятностью, время этого муравья уменьшается на величину расстояния от текущей вершины до той, в которую он перешёл. Если случайная величина больше q, то в качестве вершины перехода берётся случайная вершина, для которой верно, что вероятность перехода в неё больше 0. В этом случае точно так же время этого муравья уменьшается на величину расстояния от текущей вершины до той, в которую он перешёл.

   - Путь = [1, 2, 3, 5]
   - Длина пути = 46

7. Локально обновляем феромон на пути первого муравья по формуле:

$$\tau_{i,j}(1) = (1-p)\tau_{i,j}(0) + p\phi = (1-0.1)\cdot 10 + 0.1\cdot 10 = 0.1$$

Так как мы выбрали $\phi = n \cdot \frac{Q}{\text{age}}$, то значения остались теми же.

8. Аналогично "запустим" второго муравья. Таблица вероятностей переходов для второго муравья:

№ вершины	1	2	3	4	5
1	0	2.38095e-4	4.62963e-5	5.76701e-6	6.88705e-5
2	0	0	0.5613	0.2450	0.1344
3	0	0	0	0	0
4	0	0	2.66667e-4	0	7.55858e-5

9. Таблица случайных значений для второго муравья, на 2-ом шаге значение случайного числа превысило 0.9, значения вероятностей были нормированы и случайным образом из них было выбрано 4 значение.

   - Путь = [1, 2, 4, 3]
   - Длина пути = 38

10. Таблица сгенерированных $rand_2$:

1	2	3	4
0.794284540683907	0.913337361501670	0.337122644398882	0.414233540182197

11. Глобальное обновление феромонов не нужно, так как использовалось всего одно поколение. Алгоритм закончил свою работу.

3.5 Max-Min муравьиная система

3.5.1 Отличия Max-Min модификации от общего случая

Одной из основных проблем, с которыми сталкивается классический муравьиный алгоритм, является неэффективное использование феромонов, что может привести к преждевременному застреванию в локальных минимумах. В ответ на эту проблему была разработана Max-Min модификация муравьиного алгоритма, которая направлена на улучшение качества решений и повышение устойчивости алгоритма к застоям.

Max-Min модификация основывается на принципе ограничения феромонов, что позволяет контролировать их уровень и избегать чрезмерного накопления. В этой модификации устанавливаются максимальные и минимальные границы для значений феромонов, что обеспечивает более равномерное распределение феромонов по всем путям. Это позволяет избежать ситуации, когда один путь становится доминирующим, и способствует более разнообразному поиску решений.[1]

Причины применения Max-Min модификации разнообразны. Во-первых, она позволяет улучшить качество решений, так как феромоны обновляются более сбалансированно, что способствует более равномерному исследованию пространства решений. Во-вторых, такая модификация помогает ускорить сходимость алгоритма, так как муравьи могут более эффективно использовать информацию о феромонах, не застревая на одном и том же пути. В-третьих, Max-Min модификация снижает вероятность преждевременного застревания в локальных минимумах, так как феромоны на менее популярных путях также имеют шанс быть выбраны.[1]

Отличия Max-Min модификации от общего случая муравьиного алгоритма заключаются в подходе к обновлению феромонов. В общем случае феромоны обновляются на основе всех найденных путей, что может привести к доминированию одного или нескольких путей. В Max-Min модификации феромоны ограничиваются, что позволяет избежать чрезмерного влияния отдельных путей на выбор муравьёв. Кроме того, в Max-Min модификации феромоны обновляются с учетом как максимальных, так и минимальных значений, что обеспечивает более сбалансированное распределение феромонов.

3.5.2 Формализация Max-Min модификации

Обновление феромонов в Max-Min задается следующим образом:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{best}$$

где

$$\Delta\tau_{ij}^{best} = \frac{1}{f(s_{best})}$$

где $f(s_{best})$ обозначает длину решения, полученного либо от итерационного лучшего решения s_{ib}, либо от глобального лучшего решения s_{gb}.

В Max-Min используется только одно решение для обновления феромонов, что является важным средством для эксплуатации поиска. Выбор между итерационным лучшим решением s_{ib} и глобальным лучшим решением s_{gb} влияет на то, как история поиска будет использоваться. Использование только s_{gb} может привести к слишком быстрому сосредоточению поиска вокруг этого решения, что ограничивает исследование других возможных решений. В то время как использование s_{ib} позволяет избежать этого, так как итерационные лучшие решения могут значительно различаться от итерации к итерации.

Max-Min вводит явные ограничения на минимальные и максимальные значения феромонов:

$$\tau_{min} \leq \tau_{ij}(t) \leq \tau_{max}$$

После каждой итерации необходимо убедиться, что феромоны соответствуют этим ограничениям:

$$\text{если } \tau_{ij}(t) > \tau_{max}, \text{ то } \tau_{ij}(t) = \tau_{max}$$

$$\text{если } \tau_{ij}(t) < \tau_{min}, \text{ то } \tau_{ij}(t) = \tau_{min}$$

3.5.3 Тестовый пример для Max-Min модификации

Матрица расстояний:

$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 10 & 0 & 35 & 25 \\ 15 & 35 & 0 & 30 \\ 20 & 25 & 30 & 0 \end{bmatrix}$$

- Количество поколений: 1
- Количество муравьёв в поколении: 2
- Коэффициент запаха $\alpha = 1$
- Коэффициент расстояния $\beta = 2$
- Глобальный коэффициент обновления $e = 0.1$
- Локальный коэффициент обновления $p = 0.1$
- Количество выпускаемых феромонов $Q = 1$
- Минимальный уровень феромонов $\tau_{min} = 0.1$
- Максимальный уровень феромонов $\tau_{max} = 10$

1. Для начала проинициализируем каждый элемент матрицы феромонов начальным значением $\tau_{i,j}(0) = \phi = \frac{n \cdot Q}{2} = 1$, а затем обнулим диагональные элементы.
2. Затем для каждого из муравьёв подсчитаем вероятности перехода в k-тый город из первого города по формуле из алгоритма выше. Для тех городов, где муравей уже был, вероятность считаем равной 0.
3. Пример расчета вероятности перехода первого муравья из первого города во второй:

$$P_{1,2} = \frac{(\tau_{1,2}(t))^{\alpha} \cdot (\eta_{1,2})^{\beta}}{\sum_{l \in J_1} (\tau_{1,l}^{\alpha} \cdot \eta_{1,l}^{\beta})} = \frac{(1)^1 \cdot (1/10)^2}{(1 + 1/10^2 + 1/15^2)} \approx 0.0526$$

4. Таблица вероятностей переходов для первого муравья:

№ вершины	1	2	3	4
1	0	0.0526	0.0176	0.0128
2	0	0	0.0455	0.0333
3	0	0	0	0.0333
4	0	0	0	0

5. Таблица сгенерированных $rand_1$:

1	2	3
0.3	0.6	0.8

6. На каждой итерации сгенерируем случайное число $rand_k$, лежащее от 0 до 1. Если все вероятности перехода для данного муравья равны 0, значит, муравей закончил свой путь, и мы переходим к следующему. Иначе, если случайная величина, соответствующая муравью, меньше или равна коэффициенту $q = 0.9$, то муравей переходит в вершину с наибольшей вероятностью, время этого муравья уменьшается на величину расстояния от текущей вершины до той, в которую он перешёл. Если случайная величина больше q, то в качестве вершины перехода берётся случайная вершина, для которой верно, что вероятность перехода в неё больше 0.

   - Путь = [1, 2, 4, 3]
   - Длина пути = 65

7. Локально обновляем феромон на пути первого муравья по формуле:

$$\tau_{i,j}(1) = (1-p)\tau_{i,j}(0) + p \cdot \phi = (1-0.1) \cdot 1 + 0.1 \cdot 1 = 0.1 + 0.1 = 0.2$$

   Таким образом, феромоны на пути между городами обновляются, и их значения становятся:

$$\tau_{1,2}(1) = 0.2, \quad \tau_{2,4}(1) = 0.2, \quad \tau_{4,3}(1) = 0.2$$

8. Аналогично "запустим" второго муравья. Таблица вероятностей переходов для второго муравья:

№ вершины	1	2	3	4
1	0	0.0526	0.0176	0.0128
2	0	0	0.0455	0.0333
3	0	0	0	0.0333
4	0	0	0	0

9. Таблица случайных значений для второго муравья:

1	2	3
0.4	0.7	0.5

10. На второй итерации случайное число для второго муравья также будет сгенерировано. Если оно меньше $q = 0.9$, то муравей выбирает город с наибольшей вероятностью. В противном случае, он выбирает случайный город с ненулевой вероятностью. Предположим, что случайное число для второго муравья оказалось больше q, и он выбрал город 3.

    - Путь = [1, 3, 2, 4]
    - Длина пути = 60

11. Локально обновляем феромон на пути второго муравья по формуле:

$$\tau_{i,j}(1) = (1-p)\tau_{i,j}(0) + p \cdot \phi = (1-0.1) \cdot 1 + 0.1 \cdot 1 = 0.1 + 0.1 = 0.2$$

    Таким образом, феромоны на пути между городами обновляются, и их значения становятся:

$$\tau_{1,3}(1) = 0.2, \quad \tau_{3,2}(1) = 0.2, \quad \tau_{2,4}(1) = 0.2$$

12. Глобальное обновление феромонов происходит на основе лучшего найденного решения. Предположим, что лучшее решение было найдено первым муравьём с длиной пути 65. Обновление феромонов происходит по формуле:

$$\tau_{i,j}(t+1) = (1-e)\tau_{i,j}(t) + e \cdot \Delta\tau_{i,j}^{best}$$

    где

$$\Delta\tau_{i,j}^{best} = \frac{Q}{L_{best}} = \frac{1}{65}$$

    Таким образом, обновление феромонов для всех путей:

$$\tau_{1,2}(1) = (1-0.1) \cdot 0.2 + 0.1 \cdot \frac{1}{65} \approx 0.2$$

$$\tau_{2,4}(1) = (1 - 0.1) \cdot 0.2 + 0.1 \cdot \frac{1}{65} \approx 0.2 + 0.0001538 \approx 0.2001538$$

$$\tau_{4,3}(1) = (1 - 0.1) \cdot 0.2 + 0.1 \cdot \frac{1}{65} \approx 0.2 + 0.0001538 \approx 0.2001538$$

13. После обновления феромонов, значения феромонов на всех путях становятся:

$$\begin{bmatrix} \tau_{1,2} & \tau_{1,3} & \tau_{1,4} \\ \tau_{2,1} & \tau_{2,3} & \tau_{2,4} \\ \tau_{3,1} & \tau_{3,2} & \tau_{3,4} \\ \tau_{4,1} & \tau_{4,2} & \tau_{4,3} \end{bmatrix} = \begin{bmatrix} 0 & 0.2 & 0.2 \\ 0.2 & 0 & 0.2001538 \\ 0.2 & 0.2 & 0 \\ 0.2 & 0.2001538 & 0.2 \end{bmatrix}$$

14. Алгоритм завершает свою работу после одного поколения. Лучшее найденное решение:
    - Путь = [1, 2, 4, 3]
    - Длина пути = 65

4 Сравнительный анализ и экспериментальные результаты

Для постановки первого сравнительного эксперимента воспользуемся временным ограничением в 250 миллисекунд.

- **nodes**:
  - Список координат узлов (городов), представленный в виде кортежей.
  - Пример:
    $$\text{nodes} = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$$
    где n — количество узлов.
- **colony_size**:
  - Размер колонии муравьев (количество муравьев, участвующих в поиске решения).
  - Пример:
    $$\text{colony\_size} = 5$$
- **elitist_weight**:
  - Вес, с которым обновляются феромоны на лучшем найденном маршруте в элитарном алгоритме.
  - Пример:
    $$\text{elitist\_weight} = 1.0$$
- **min_scaling_factor**:
  - Минимальный коэффициент масштабирования для феромонов в Max-Min алгоритме.
  - Пример:
    $$\text{min\_scaling\_factor} = 0.001$$
- **alpha**:
  - Коэффициент, контролирующий влияние феромонов на выбор следующего узла.
  - Пример:
    $$\alpha = 1.0$$
- **beta**:
  - Коэффициент, контролирующий влияние эвристической информации на выбор следующего узла.
  - Пример:
    $$\beta = 3.0$$

- **rho**:
  - Коэффициент испарения феромонов.
  - Пример:
$$\rho = 0.1$$
- **phi**:
  - Коэффициент, используемый для локального обновления феромонов.
  - Пример:
$$\phi = 0.1$$
- **pheromone_deposit_weight**:
  - Вес, с которым феромоны добавляются на рёбра, пройденные муравьями.
  - Пример:
$$\text{pheromone\_deposit\_weight} = 1.0$$
- **initial_pheromone**:
  - Начальное значение феромонов на рёбрах.
  - Пример:
$$\text{initial\_pheromone} = 1.0$$
- **steps**:
  - Количество итераций (шагов), которые будет выполнять алгоритм для поиска решения.
  - Пример:
$$\text{steps} = 100$$

В основном блоке кода, который запускает алгоритм, параметры задаются следующим образом:

```
_colony_size = 5
_steps = 200
random.seed(10)
_nodes = [(random.uniform(0, 200), random.uniform(0, 200)) for _ in range(0, 50)]
```

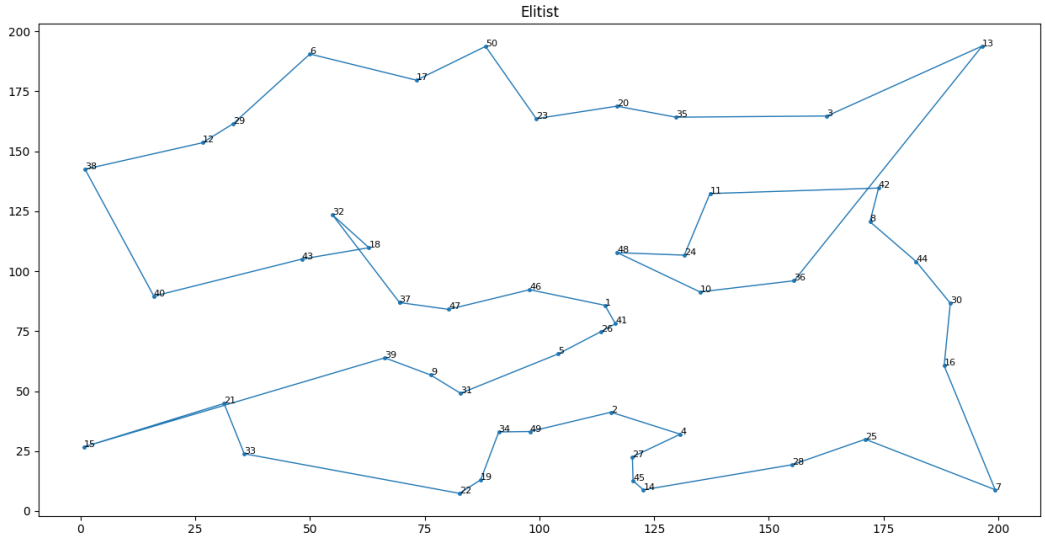

Figure 1: Маршрут, полученный с помощью элитарной модификации

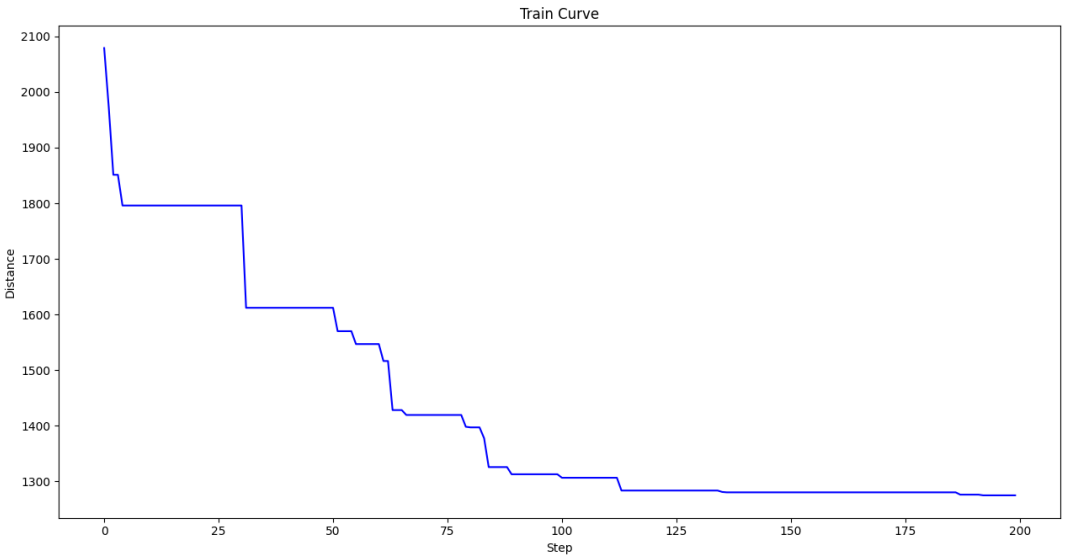

Figure 2: Зависимость расстояния от количества итераций для элитарной модификации

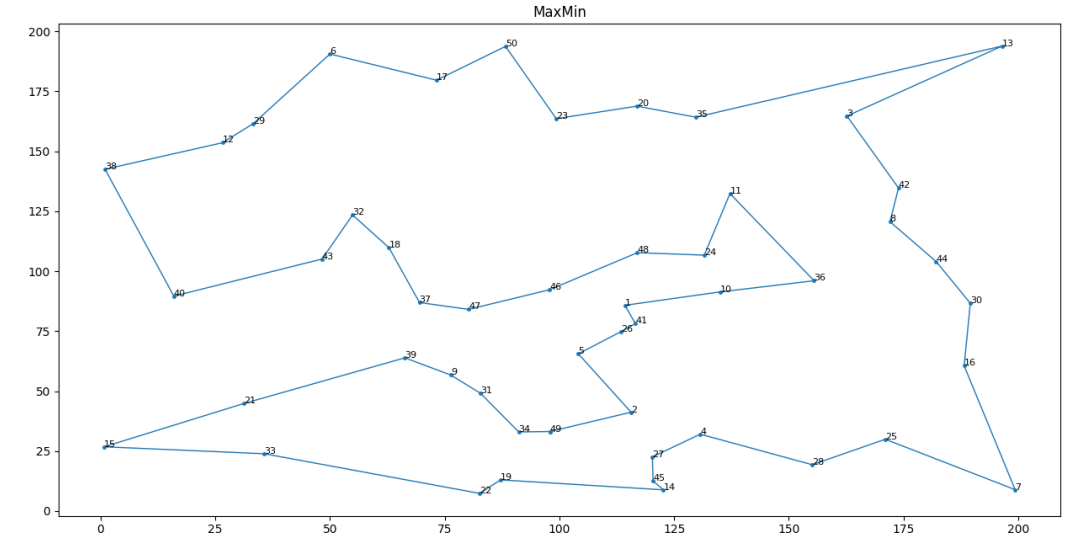

Figure 3: Маршрут, полученный с помощью Max-Min модификации

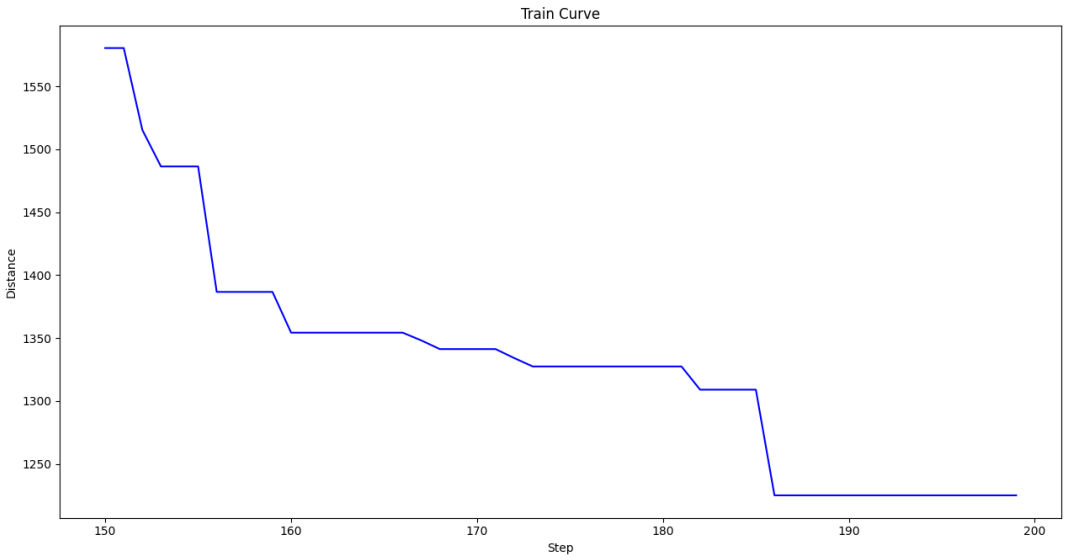

Figure 4: Зависимость расстояния от количества итераций для Max-Min модификации

```
Started : Elitist
Ended : Elitist
Sequence : 37 -> 47 -> 46 -> 1 -> 41 -> 26 -> 5 -> 31 -> 9 -> 39 -> 15 -> 21 -> 33 -> 22 -> 19 -> 34 -> 49 -> 2 -> 4 -> 27 -> 45 -> 14 -> 28 -> 25 -> 7 -> 1
6 -> 30 -> 44 -> 8 -> 42 -> 11 -> 24 -> 48 -> 10 -> 36 -> 13 -> 3 -> 35 -> 20 -> 23 -> 50 -> 17 -> 6 -> 29 -> 12 -> 38 -> 40 -> 43 -> 18 -> 32
Total distance travelled to complete the tour : 1274.95

Started : MaxMin
Ended : MaxMin
Sequence : 49 -> 34 -> 31 -> 9 -> 39 -> 21 -> 15 -> 33 -> 22 -> 19 -> 14 -> 45 -> 27 -> 4 -> 28 -> 25 -> 7 -> 16 -> 30 -> 44 -> 8 -> 42 -> 3 -> 13 -> 35 ->
20 -> 23 -> 50 -> 17 -> 6 -> 29 -> 12 -> 38 -> 40 -> 43 -> 32 -> 18 -> 37 -> 47 -> 46 -> 48 -> 24 -> 11 -> 36 -> 10 -> 1 -> 41 -> 26 -> 5 -> 2
Total distance travelled to complete the tour : 1225.1

Press any key to continue . . .
```

Figure 5: Пример выходных данных

- EAS может быстро находить хорошие решения, особенно на простых экземплярах TSP, благодаря использованию информации о всех муравьях. Однако, на более сложных экземплярах EAS может не всегда находить глобальный оптимум из-за риска застревания в локальных минимумах.[4]
- EAS может демонстрировать высокую эффективность на малых и средних задачах, но ее производительность может ухудшаться на больших и сложных экземплярах.[4]
- MMAS, благодаря своей структуре обновления феромонов, часто показывает более высокую эффективность в нахождении глобального оптимума, особенно на сложных экземплярах TSP. Ограничение феромонов помогает избежать чрезмерного накопления феромонов на менее оптимальных путях, что способствует более разнообразному поиску.
- MMAS может требовать больше итераций для достижения хороших решений, но в конечном итоге часто находит более качественные решения.

Из полученных результатов очевидно, что при данной постановке с большим количеством итераций для задачи средней сложности Max-Min модификация справляется лучше: этот алгоритм находит более короткий маршрут за отведённое время, и, очевидно, лучше сходится к оптимуму.

Теперь упростим муравьям задачу, сократив количество вершин графа и оставив 20 итераций для нахождения результата.

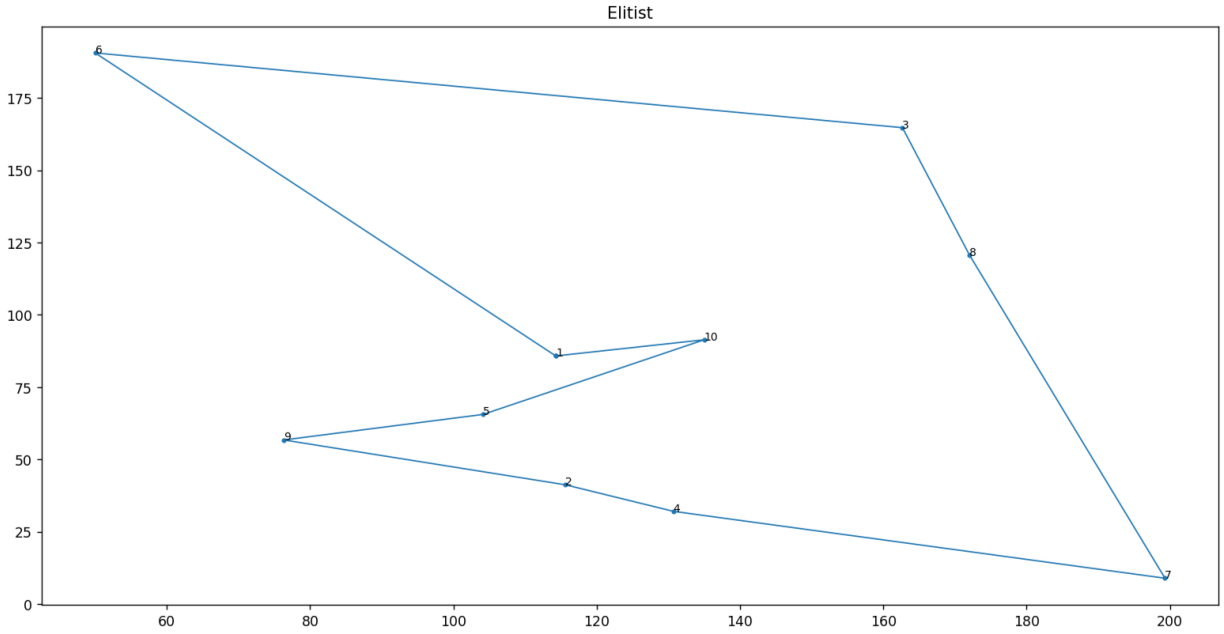

Figure 6: Маршрут, полученный с помощью элитарной модификации

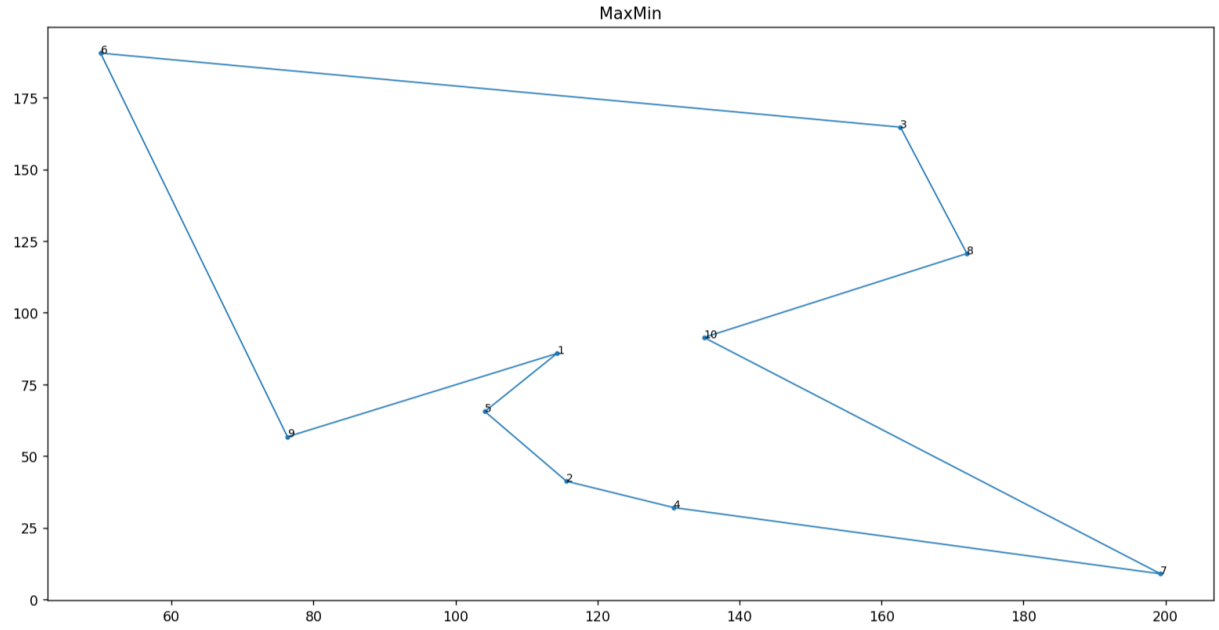

Figure 7: Маршрут, полученный с помощью Max-Min модификации

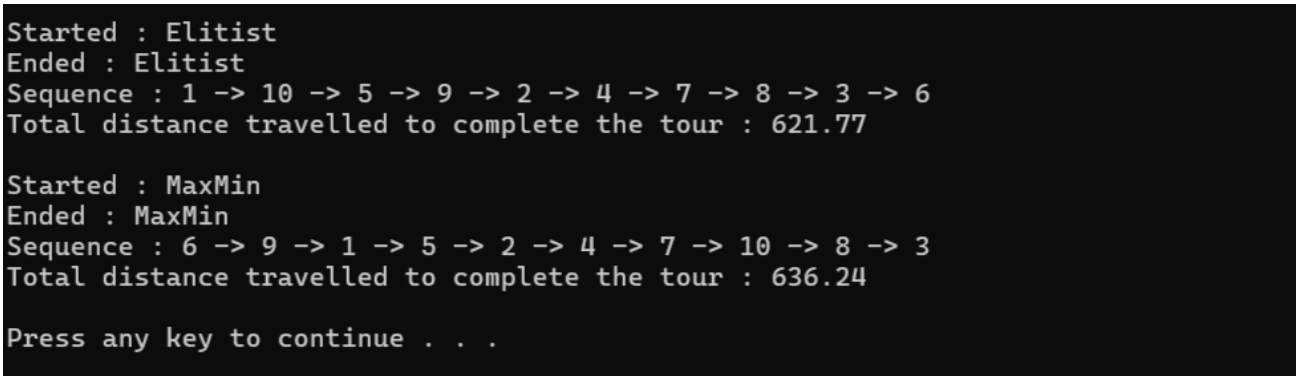

Figure 8: Пример выходных данных

Очевидно, на более простой задаче элитарная модификация показывает лучший результат, причём с существенным отрывом, что подтверждает наши теоретические данные.

5 Заключение

В ходе данной курсовой работы был проведен сравнительный анализ двух модификаций муравьиного алгоритма — элитарной модификации (EAS) и Max-Min модификации (MMAS) — в контексте решения задачи коммивояжера (TSP). Оба подхода продемонстрировали свою эффективность в поиске оптимальных решений, однако их характеристики и поведение в процессе поиска существенно различаются.

Элитарная модификация муравьиного алгоритма, благодаря использованию информации о всех найденных решениях, позволяет быстро находить хорошие маршруты, особенно на простых и средних экземплярах задачи. Однако, ее склонность к застреванию в локальных минимумах может ограничивать эффективность на более сложных задачах. Это делает EAS подходящим выбором для ситуаций, где требуется быстрое получение приемлемых решений, но не всегда гарантирует нахождение глобального оптимума.

С другой стороны, Max-Min модификация муравьиного алгоритма, с ее строгими ограничениями на феромоны и обновлением только на основе лучших найденных решений, демонстрирует более стабильную и надежную сходимость к глобальному оптимуму. Хотя MMAS может требовать больше итераций для достижения хороших результатов, его способность избегать локальных минимумов делает его более предпочтительным для сложных задач, где качество решения имеет первостепенное значение.

В результате проведенного анализа можно сделать вывод, что выбор между EAS и MMAS должен основываться на конкретных требованиях задачи. Для задач, где важна скорость нахождения решения, EAS может быть более подходящим вариантом. В то время как для задач, требующих высокой точности и надежности, MMAS представляется более эффективным выбором.

6 Использованные источники

1. Stützle, T., Hoos, H. H. (2000). MAX–MIN Ant System. IRIDIA, Université Libre de Bruxelles. С. 892; С. 898-905

2. Negulescu, S. C., Oprean, C., Kifor, C. V., Carabulea, I. (2008). Elitist ant system for route allocation problem. Faculty of Engineering, “Lucian Blaga” University of Sibiu. — С. 63-65

3. Dorigo, M., Birattari, M., Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. Université Libre de Bruxelles, Belgium. — С. 30-32

4. Павленко, А. И., Титов, Ю. П. (2023). Сравнительный анализ модифицированных методов муравьиных колоний. Московский авиационный институт. — С. 102-105