УДК 656.02 DOI: 10.31799/2077-5687-2023-4-47-53

ИССЛЕДОВАНИЕ АЛГОРИТМА МАРШРУТИЗАЦИИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА НА ПРИМЕРЕ АЛГОРИТМА МУРАВЬИНОЙ КОЛОНИИ

Д. В. Кучко, А. С. Костин
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Данная статья представляет анализ муравьиного алгоритма для решения задачи коммивояжера. Данный алгоритм является одним из ключевых оптимизационных алгоритмов. Поведение муравьев в задачи поиска оптимального маршрута к пище является одним из поводов для применения в различных областях, таких как маршрутизация сетей, комбинаторная оптимизация, задачи планирования, и других задачах, связанных с оптимизацией процессов.

Ключевые слова: алгоритм маршрутизации, задача коммивояжера, муравьиный алгоритм, задачи оптимизации.

Для цитирования:
Кучко, Д. В. Исследование алгоритма маршрутизации для решения задачи коммивояжера на примере алгоритма муравьиной колонии / Д. В. Кучко, А. С. Костин // Системный анализ и логистика. – 2023. – № 4(38). – с. 47 – 53. DOI: 10.31799/2077-5687-2023-4-47-53.

RESEARCH OF A ROUTING ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM USING THE EXAMPLE OF THE ANT COLONY ALGORITHM

D. V. Kuchko, A. S. Kostin
St. Petersburg State University of Aerospace Instrumentation

This article presents an analysis of the ant algorithm for solving the traveling salesman problem. This algorithm is one of the key optimization algorithms. The behavior of ants in the problem of finding the optimal route to food is one of the reasons for application in various fields, such as network routing, combinatorial optimization, scheduling problems, and other problems related to process optimization.

Keywords: routing algorithm, traveling salesman problem, ant algorithm, optimization problems.

For citation:
Kuchko, D. V. Research of a routing algorithm for solving the traveling salesman problem using the example of the ant colony algorithm / D. V. Kuchko, A. S. Kostin // System analysis and logistics. – 2023. – № 4(38). – p. 47 – 53. DOI: 10.31799/2077-5687-2023-4-47-53.

Введение

С развитием технологий, в том числе и с появлением новых средств доставки грузов, таких как беспилотные авиационные системы, возникает потребность в оптимизации транспортных процессов [1,2]. Вопросы, связанные с организацией перевозок, послужили для выделения отдельного класса оптимизационных задач - транспортных задач. Одной из таких задач является известная задача коммивояжера. До определенного момента считалось, что решение данной задачи возможно только методом перебора всех возможных вариантов, однако, на данный момент, для решения задачи все больше используется различные алгоритмы глобального планирования траектории [3]. Задача коммивояжера заключается в следующем: отыскать кратчайший путь в полном конечном графе с N вершинами. Для решения данной задачи возможно применение различных алгоритмов планирования траектории, все основные алгоритмы делятся на традиционные и интеллектуальные алгоритмы (рис. 1).

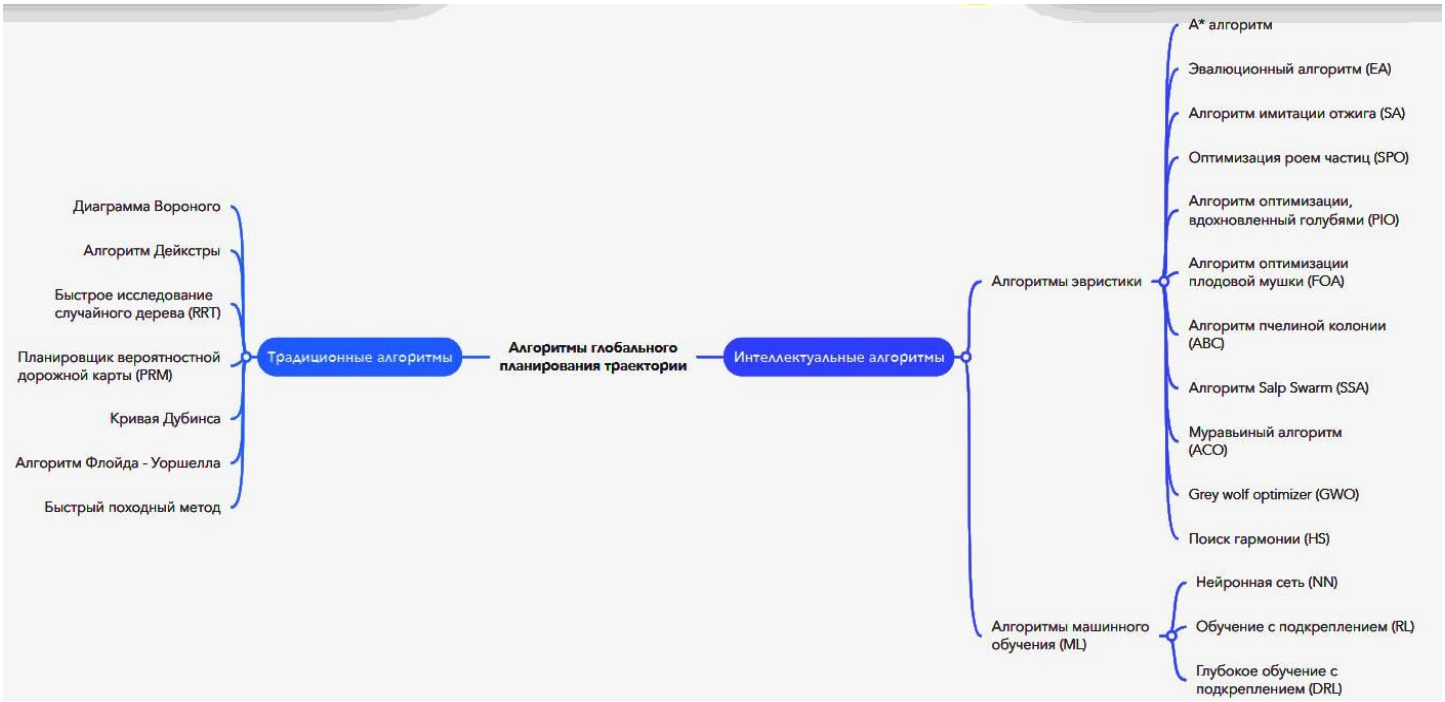

Рис. 1. Классификация оптимизационных алгоритмов

Большинство интеллектуальных алгоритмов — это оптимизационные алгоритмы, осуществляющие поиск приближенных допустимых решений на основе определенных свойств для нахождения решения. Это не систематический поиск, а использование предыдущих итераций для выбора эффективных методов, и он не может гарантировать высокую скорость нахождения решений и высокую степень оптимальности полученных решений [4]. К эвристическим алгоритмам, используемых для нахождения оптимального маршрута БАС, относятся следующие алгоритмы: алгоритм имитации отжига (SA), эволюционный алгоритм (EA), метод роя частиц (PSO), алгоритм оптимизации на основе муравьиной колонии (ACO) и другие. Далее рассмотрим алгоритм муравьиной колонии более подробно.

Муравьиный алгоритм

Муравьиный алгоритм основан на наблюдении за муравьями и их способности находить кратчайшие пути до их цели. Пока муравей движется, он оставляет за собой след из феромонов, который, в свою очередь рассеивается со временем. Феромон – химическое вещество, выделяемое муравьем, помогающее в взаимодействии между отдельными особями. В зависимости от количества муравьев, расстояния до цели и времени интенсивность феромонов изменяется. Чем больше муравьев, тем интенсивность больше. Также расстояние и временя, затраченное на путь влияет на феромоновый след: чем расстояние больше, тем больше вероятность что феромон испарится [5].

Таким образом, можно сказать, что в зависимости от информации, полученной от следа одной особи другою и работает данный алгоритм.

В основе решения задачи коммивояжера лежит вышеописанное поведение муравьев. Данный алгоритм позволяет с большой точностью и относительно большой скоростью найти оптимальное решение [6].

Математическое описание метода

Рассмотрим вариант задачи поиска кратчайшего пути между N вершинами. Вероятность перемещения отдельного муравья из одной вершины (i) в другую (j) определяется близостью (η_{ij}) между вершинами и количеством феромонов (τ_{ij}), оставленных другими муравьями.

То есть вероятность перемещения из i вершины в j будет определяться формулой:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum \tau_{ij}^{\alpha} \eta_{ij}^{\beta}}, \quad (1)$$

где α и β - настраиваемые параметры алгоритма, позволяющие определить влияние близости и опыта прошлых особей на выбор муравья.

При α = 0 муравей не будет учитывать опыт прошлых особей, и его выбор будет зависеть только от близости между вершинами. При β = 0 ситуация будет обратной: муравей будет учитывать опыт прошлых особей и не будет учитывать близость между вершинами. В случае обнуления параметра а данный алгоритм приравнивается к жадным алгоритмам, а в случае обнуления β алгоритм приведет к быстрому нахождению субоптимального решения.

Для определения количества феромона на ребрах маршрута из вершины i в вершину j, мы используем следующие формулы:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, если (i,j) \in T_k(t) \\ 0, если (i,j) \notin T_k(t) \end{cases}, \quad (2)$$

где $\Delta_{\tau ij,k}(t)$ - добавка феромона на грань (i, j) для муравья k в итерации t, Q - константа, регулируемая в процессе эксперимента., L_k - длина маршрута, пройденного муравьем k.

Для расчета количества феромона на новой итерации (t + 1) на грани (i, j):

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij,k}(t) \quad (3)$$

где $\tau_{ij}(t+1)$ - количество феромона на грани (i, j) в новой итерации (t + 1);

$\tau_{ij}(t)$ - количество феромона на грани (i, j) в предыдущей итерации (t);

ρ - коэффициент испарения феромона, который обычно находится в диапазоне от 0 до 1.

Реализация алгоритма

Для получения автоматизированного решения задачи разработана программа на высокоуровневом языке программирования Python.

Данная реализация позволяет проводить анализ решения при различных комбинациях настраиваемых параметров с последующей визуализацией решений [7].

В качестве исходных данных подаётся массив, состоящий из координат исследуемых точек.

Происходит вычисление расстояний, в случае если они не обозначены изначально, между точками по формуле $\sqrt{((x_2-x_1)^2 + (y_2-y_1)^2)}$.

Относительно настроек программы происходит заполнение массива феромона.

Затем начинается размещение муравьёв в вершинах графа. В данной версии движение начинается всегда с 0-й вершины. Затем муравьи начинают направление, определяющееся вероятностным методом, на основании формулы 1:

```
def PPerehoda(k, TownList):
    global Summa
    for i in range(len(TownList)):
        r = int(TownList[i])
        if k != r:
            Summa = Summa + ((Feramonus[k][r] ** Alfa) * ((Const/RastBtw[k][r]) ** Beta))
```

```
        for i in range(len(TownList)):
            r = int(TownList[i])
            if k != r:
                PMove[k][r] = ((Feramonus[k][r] ** Alfa) * ((Const/RastBtw[k][r]) ** Beta)) /
Summa
```

После прохода каждого муравья происходит испарение феромона и добавление его на каждый участок пути. Количество феромона на участок вычисляется по формуле 2:

```
    for i in range(CountTown):
        XCoord = np.append(XCoord,XYCoord[i][0])
        YCoord = np.append(YCoord,XYCoord[i][1])
    for i in range(len(XYCoord)):
        for j in range(len(XYCoord)):
            if i != j:
                Feramonus[i][j] = CountFeramonus
    for i in range(len(XYCoord)):
        TownListN = np.append(TownListN,i)
    for i in range(len(XYCoord)):
        for j in range(len(XYCoord)):
            RastBtw[i][j] = round(math.sqrt(((XYCoord[j][0]-XYCoord[i][0])**2) +
((XYCoord[j][1]-XYCoord[i][1])**2)), 3)
    for q in range(CountGlobalIter):
        plt.xlim(0,int(max(XCoord)+1))
        plt.ylim(0,int(max(YCoord)+1))
        plt.scatter(XCoord, YCoord, color = 'black')
        plt.pause(0.0001)
        for i in range(CountIter):
            NextTown = k
            TownList = np.copy(TownListN)

            while len(TownList) > 0:
                k = NextTown
                PPerehoda(k,TownList)
                RandomTown(k,TownList)
                TownList = np.delete(TownList,TownList.tolist().index(k))
                TownListFinale = np.append(TownListFinale,k)
            Visual(TownListFinale, 200, 0)
             Travel = 0
            for i  in  range(len(TownListFinale)-1):
                j = int(TownListFinale[i])
                k = int(TownListFinale[i+1])
                Travel += RastBtw[j][k]
            Travel += RastBtw[int(TownListFinale[0])][int(TownListFinale[-1])]
            RezRastMarsh = np.append(RezRastMarsh, Travel)
            RezRastMarshPath = np.append(RezRastMarshPath, TownListFinale)
            Travel = []
            TownListFinale = []
            PMove = np.zeros([len(XYCoord), len(XYCoord)])
        RezRastMarshPath = np.split(RezRastMarshPath,CountIter)
        Feramonus = np.multiply(Feramonus, CoefIspareniya)
```

```
for i in range(CountIter):
    for j in range(len(RezRastMarshPath[i])-1):
        k = int(RezRastMarshPath[i][j])     # i В матрице феромонов
        o = int(RezRastMarshPath[i][j+1])   # j В матрице феромонов
        Feramonus[k][o] += ConstForFeramonus / RezRastMarsh[i]
MinForIter = np.append(MinForIter, min(RezRastMarsh))
MinIndex = RezRastMarsh.tolist().index(min(RezRastMarsh))
MinForIterPath = np.append(MinForIterPath, RezRastMarshPath[MinIndex])
RezRastMarsh = []
RezRastMarshPath = []
plt.cla()
```

Итогом является обновленный массив феромонов. С каждой новой итерацией муравьи будут строить маршрут, в большей степени, ориентируясь по тем тропам, на которых феромона оказалось больше.

В результате работы программы формируется массив наименьших по расстоянию путей на каждой итерации и выводится кротчайший. Так же этот маршрут отображается графически. Результаты при различном количестве вершин графа представлены на рисунке 2.

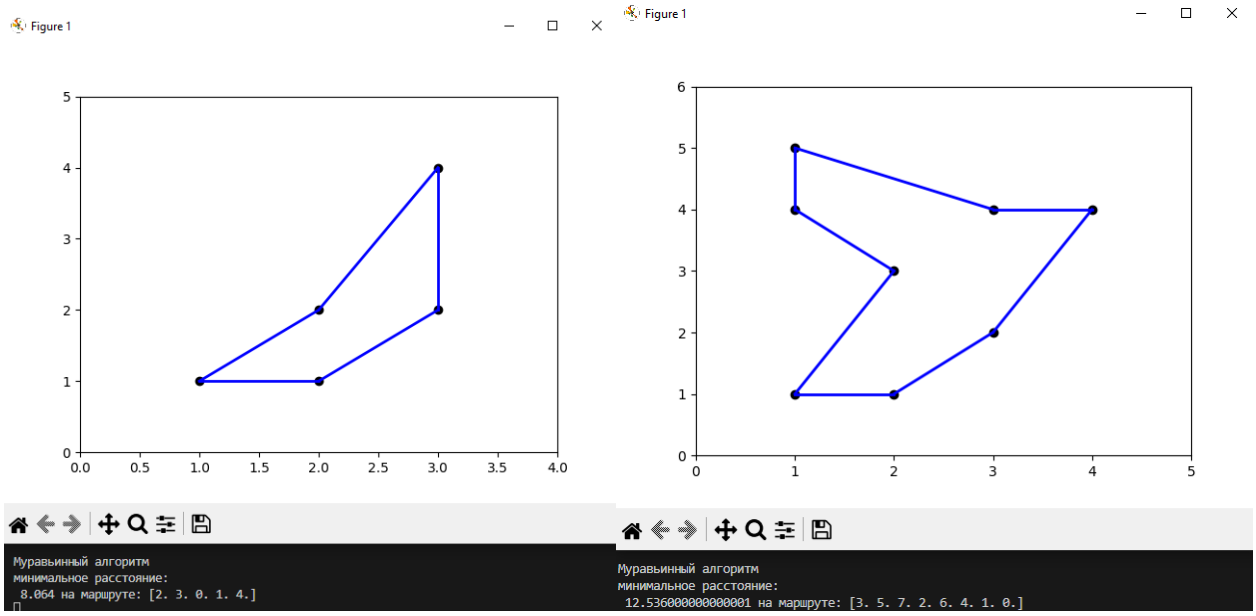

Рис. 2. Результат работы алгоритма

В результате работы программы формируется вариант маршрута движения беспилотной системы на основе исходных данных. Данный алгоритм возможно модернизировать с целью ускорения поиска решения задачи за счет ввода дополнительных условий, например «муравьев», которые двигаются по всем ребрам графа случайным образов, без учета феромона, также данный алгоритм позволяет построить маршрут с учетом введенных ограничений и запрещающих ребер графа, по которым невозможно произвести движение, за счет чего моделируются различные сценарии движения в условиях изменчивой внешней среды.

Например, в статье [8] рассмотрен распределенный алгоритм поиска маршрута движения на основе двойственной декомпозиции цепей связи дронов. Данный вариант реализации метода улучшает традиционный алгоритм оптимизации муравьиной колонии с точки зрения выбора траектории, обновления феромонов и решает проблемы низкой скорости поиска решения и адаптивности алгоритма муравьиной колонии.

Заключение

В данной статье проведена работа по исследованию методов для решения задачи коммивояжера. Выполнена программная реализация для решения данной задачи с помощью муравьиного алгоритма. Построен путь перемещения между вершинами графа.

Использование вышеописанного алгоритма для решения задачи демонстрирует его эффективность. Данная реализация не раскрывает полноту математической модели в полной мере, однако данный алгоритм имеет возможности модернизации, такие как изменение начальной точки отправления, а также внедрение «элитных» муравьев, что поспособствует повышению точности и качества оптимизационной задачи.

СПИСОК ЛИТЕРАТУРЫ

1. Костин, А. С. Разработка автоматизированных решений для исследования вариантов маршрутов доставки при совместном использовании транспортного средства и беспилотной авиационной системы в границах города / Н. Н. Майоров, А. С. Костин // Известия Тульского государственного университета. Технические науки. – 2022. – № 7. – С. 348-356.
2. Костин, А. С. Исследование моделей и методов маршрутизации практического выполнения автономного движения беспилотными транспортными системами для доставки грузов / А. С. Костин, Н. Н. Майоров // Вестник государственного университета морского и речного флота имени адмирала С.О. Макарова. – 2023. – Т. 15. – № 3. – С. 524- 536.
3. Калиберда, Е. А. Муравьиный алгоритм в решении задачи коммивояжера / Е. А. Калиберда, М. Ю. Гуненков, Д. С. Дюсекенов, И. В. Федотова // Прикладная математика и фундаментальная информатика. – 2020. – Т. 7, № 2. – С. 10-17.
4. Коцюбинская, С. А. Задача глобальной оптимизации. Муравьиный алгоритм / С. А. Коцюбинская // Modern Science. – 2020. – № 5-1. – С. 537-540.
5. Yunhong, Y. UAV Formation Trajectory Planning Algorithms: A Review [Электронный ресурс] / Y. Yunhong, X. Xingzhong, Y. Yuehao // Drones. – 2023. – Vol. 7. – P. 45. – URL: https://www.researchgate.net/publication/367183373_UAV_Formation_Trajectory_Planning_Algorithms_A_Review (дата обращения: 15.11.2023).
6. Фаттахов, М. Р. Рынок беспилотных авиационных систем в России: состояние и особенности функционирования в макроэкономических условиях 2022 года / М. Р. Фаттахов, А. В. Киреев, В. С. Клещ // Вопросы инновационной экономики. – 2022. – Том 12. – № 4. – С. 2507–2528.
7. Свидетельство о государственной регистрации программы для ЭВМ № 2023612873. РФ. Программа формирования допустимого маршрута перемещения задачи коммивояжера на основе муравьиного алгоритма / А. С. Костин, Д. В. Кучко; правообладатель ФГАО ВО ГУАП (RU). Опубл. 08.02.2023, Реестр программ для ЭВМ. – 1 с.
8. Wei, X. Distributed path planning of unmanned aerial vehicle communication chain based on dual decomposition / X. Wei, J. Xu // Wireless Communications and Mobile Computing. – 2021. – Vol. 2021. – № 2. – P. 1-12. – DOI: 10.1155/2021/6661926.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Кучко Дмитрий Витальевич –
Студент кафедры системного анализа и логистики
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: dmitriy.kuchko@mail.ru

Костин Антон Сергеевич –
Ассистент кафедры системного анализа и логистики
Санкт-Петербургский государственный университет аэрокосмического приборостроения
190000, Санкт-Петербург, ул. Большая Морская, д. 67, лит. А
E-mail: anton13258@mail.ru

INFORMATION ABOUT THE AUTHORS

Kuchko Dmitry Vitalievich –
Student at the Department of System Analysis and Logistics
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: dmitriy.kuchko@mail.ru

Kostin Anton Sergeevich –
Assistant of the Department of System Analysis and Logistics
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: anton13258@mail.ru