НАСТАВНИК-УЧЕНИК

TUTOR-FOLLOWER

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика. 2024.
Выпуск 2 (51)
Bulletin of Syktyvkar University.
Series 1: Mathematics. Mechanics. Informatics. 2024; 2 (51)

Научная статья

УДК 004.021,630.431
https://doi.org/10.34130/1992-2752_2024_2_73

МЕТАЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА. БИБЛИОТЕКА PYTHON SCIKIT-OPT

Надежда Николаевна Бабикова, Михаил Михайлович Глухой, Евгения Николаевна Старцева, Никита Адрианович Чернян
Сыктывкарский государственный университет
имени Питирима Сорокина, valmasha@mail.ru

Аннотация. В статье рассматриваются метаэвристические методы решения задачи коммивояжера. Представлены результаты тестирования алгоритма муравьиной колонии и генетического алгоритма библиотеки Python Scikit-opt на двух наборах данных (бенчмарках) популярной и широко используемой библиотеки TSPLIB. Тестирование показало возможность применения библиотеки на практике и в учебном процессе: за приемлемое время получены решения, близкие к оптимальным.

Ключевые слова: Python, Scikit-opt, TSPLIB, генетический алгоритм, алгоритм муравьиной колонии

Для цитирования: Бабикова Н. Н., Глухой М. М., Старцева Е. Н., Чернян Н. А. Метаэвристические алгоритмы решения задачи коммивояжера. Библиотека Python Scikit-opt // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика.* 2024. Вып. 2 (51). С. 73–88. https://doi.org/10.34130/1992-2752_2024_2_73

Article

Metaheuristic algorithms for the traveling salesman problem. Python library Scikit-opt

Nadezhda N. Babikova, Mikhail M. Glukhoy, Evgenija N. Startseva, Nikita A. Chernyan
Pitirim Sorokin Syktyvkar State University, valmasha@mail.ru

Abstract. The article discusses metaheuristic methods for solving the traveling salesman problem. The results of testing the ant colony algorithm and the genetic algorithm of the Python Scikit-opt library on two data sets (benchmarks) of the popular and widely used TSPLIB library are presented. Testing showed the possibility of using the library in practice and in the educational process: solutions close to optimal were obtained in an acceptable time.

Keywords: Python, Scikit-opt, TSPLIB, genetic algorithm, ant colony algorithm

For citation: Babikova N. N., Glukhoy M. M., Startseva E. N., Chernyan N. A. Metaheuristic algorithms for the traveling salesman problem. Python library Scikit-opt. *Vestnik Syktyvkarskogo universiteta. Seriya 1: Matematika. Mekhanika. Informatika* [Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Informatics], 2024, no 2 (51), pp. 73–88. (In Russ.) https://doi.org/10.34130/1992-2752_2024_2_73

1. Введение

Задача коммивояжера (Traveling Salesman Problem, TSP) – известная и популярная задача комбинаторной оптимизации: требуется найти самый короткий (выгодный) замкнутый маршрут, который позволит коммивояжеру посетить каждый город из заданного списка ровно один раз. Эквивалентная формулировка с точки зрения теории графов: дан полный взвешенный граф (где вершины представляют собой города, ребра – дороги, а веса – стоимость или протяженность этих дорог), требуется найти гамильтонов цикл с наименьшим весом.

Задача коммивояжера (ЗК) относится к числу трансвычислительных. Трансвычислительной называют задачу, для решения которой требуется обработать более 10^{93} бит информации. Если представить гипотетический компьютер, масса которого равна массе Земли, работающий с максимально возможной вычислительной скоростью (предел Бремерманна), то он обработает 10^{93} бит за 4.54 миллиарда лет

(время существования Земли) [1]. Точное решение задачи коммивояжера методом полного перебора требует проверки $(n-1)!/2$ вариантов. При числе городов, большем 66 (поскольку $66! \approx 5.443449391 \times 10^{92}$, а $67! \approx 3.647111092 \times 10^{94}$), для точного решения ЗК потребуются миллиарды лет [2].

Для задачи коммивояжера разработано большое число приближенных методов решения, в том числе метаэвристических. Метаэвристические алгоритмы реализуют прямой стохастический поиск решений, оптимальных или близких к оптимальным, пока не будет достигнуто заданное число итераций или выполнено некоторое условие [3]. Метаэвристические алгоритмы не гарантируют обнаружения точного решения задачи, но позволяют найти достаточно хорошее решение за приемлемое время [4]. Метаэвристические методы можно разделить на четыре группы: мультистартовые; методы, имитирующие физические процессы; методы «роевого» интеллекта, в том числе алгоритм муравьиной колонии; эволюционные, в том числе генетический алгоритм [5]. Генетический и муравьиный алгоритмы являются наиболее популярными метаэвристическими алгоритмами для решения ЗК.

Для оценки эффективности реализации приближенных алгоритмов можно использовать бенчмарки — эталонные наборы данных, для которых известны точные решения. Популярной и широко используемой библиотекой наборов данных для задачи коммивояжера является библиотека TSPLIB [6].

Среди более чем 100000 библиотек языка Python есть и библиотеки метаэвристических алгоритмов, например, библиотеки Aco_routing [7], ACO [8] реализуют алгоритм муравьиной колонии, библиотеки Pygad [9], DEAP [10] — генетический алгоритм. В библиотеке Scikit-opt [11] реализованы алгоритм роя частиц, алгоритм имитации отжига, алгоритм искусственной иммунной системы, алгоритм искусственной стаи рыб, *генетический алгоритм* и *алгоритм муравьиной колонии*.

Не удалось обнаружить публикаций, связанных с библиотекой Python Scikit-opt (запрос «Scikit-opt» в библиотеке Elibrary, поиск в полном тексте публикаций, в ключевых словах, в названии, в аннотации). Поэтому для оценки возможности использования алгоритма колонии муравьев и генетического алгоритма библиотеки Scikit-opt на практике и в учебном процессе было проведено тестирование этих алгоритмов на двух наборах данных библиотеки TSPLIB для симметричной задачи коммивояжера.

2. Материалы и методы

TSPLIB — это библиотека бенчмарков для задачи коммивояжера и связанных с ней проблем, впервые опубликованная Gerhard Reinelt (Гейдельберский университет) в 1991 году [12]. Файлы с наборами данных доступны на сайте Гейдельберского университета [13] в специальном формате «.tsp». Файлы можно открыть обычным текстовым редактором Блокнот (рис. 1) или обработать с помощью Python библиотеки tsplib95 (листинг 1) [14]. Каждый файл содержит спецификацию набора и сам набор данных.

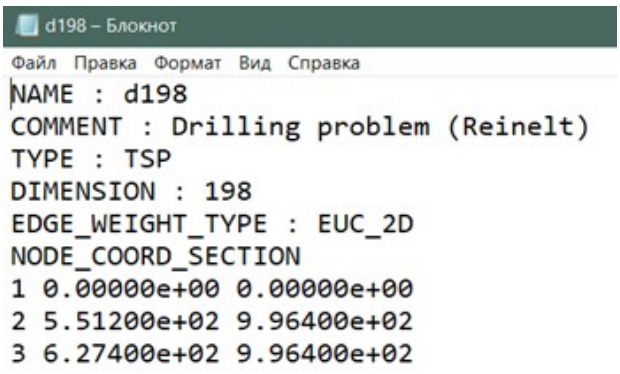

Рис. 1. Фрагмент набора данных d198

Листинг 1

Пример применения библиотеки tsplib95 для чтения набора данных библиотеки TSPLIB

```
import pandas as pd
import tsplib95
from scipy.spatial.distance import pdist, squareform
#считывание файла с расширением .tsp
with open("d198.tsp") as f:
    text = f.read()
#получение датасета из строки
problem = tsplib95.parse(text)
#извлечение координат
arr = []
for _, value in problem.node_coords.items():
    arr.append(value)
```

Окончание листинга 1

```
#расчет матрицы расстояний
dist_matrix = squareform(pdist(arr, "euclidean"))
#запись в excel файл
df1 = pd.DataFrame(dist_matrix)
df1.to_excel("d198.xlsx", header=False, index=False)
```

Тестирование проводилось на двух наборах данных:

- d198.tsp — AMD Ryzen 5 5600 6-Core Processor 3.50 GHz, G.Skill RIPJAWS V 16 ГБ 3600 МГц DDR4 Memory, среда PyCharm;
- bays29.tsp — Intel Core i5-5200U (ноутбук), 4GB DDR3 L Memory, среда IDLE.

Файл d198.tsp содержит координаты 198 вершин графа для задачи бурения, расстояние между вершинами — евклидово на плоскости. В файле bays29.tsp имеется полная матрица расстояний для 29 городов Баварии, расстояние между городами — расстояние городских кварталов.

Алгоритм муравьиных колоний имитирует механизм поведения муравьев в природе, основанный на непрямой передаче информации посредством особого пахучего вещества — феромона. Муравьи оставляют на земле следы феромонов, а другие муравьи могут почувствовать их интенсивность. Когда тропа какое-то время не используется, след феромона медленно исчезает. Чем сильнее концентрация феромонов на тропе, тем более привлекательна она для муравьев. Чем короче тропа, тем быстрее муравьи ее могут пройти и оставить новые феромоновые следы, а значит, на коротких тропах концентрация феромона будет выше. Таким образом муравьи находят короткие маршруты.

Работа алгоритма начинается с того, что некоторое количество «муравьев» помещается в вершины графа. Каждый «муравей» выбирает следующую вершину для перемещения случайным образом: если «муравей» находится в i вершине, то вероятность перемещения в j вершину вычисляется по формуле

$$p_{ij} = \frac{(\tau_{ij})^{\alpha}(1/d_{ij})^{\beta}}{\sum\limits_{k \in N}(\tau_{ik})^{\alpha}(1/d_{ik})^{\beta}},$$

где τ_{ij} — количество феромона на ребре (i, j);
d_{ij} — длина ребра (i, j);
α, β — регулируемые параметры, определяющие относительное влияние феромона и длины ребра на привлекательность ребра (i, j);
N — множество вершин, которые «муравей» еще не посетил.

При $\alpha = 0$ алгоритм превращается в классический жадный алгоритм, а при $\beta = 0$ длина ребра не влияет на выбор следующей вершины. Формула определяет вероятности для каждой непосещенной вершины, но выбор вершины «муравьем» происходит по принципу «колеса рулетки»: каждая вершина имеет свой сектор с площадью, пропорциональной вероятности. Для выбора вершины нужно «бросить шарик» — сгенерировать случайное число от 0 до 1 и определить сектор, т. е. номер вершины (рис. 2).

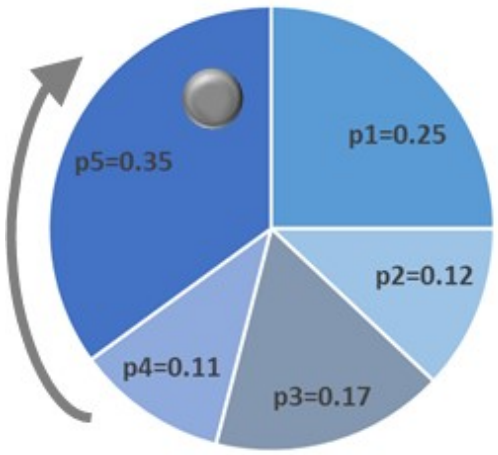

Рис. 2. Колесо рулетки

После того как все «муравьи» построят свои маршруты, определяется и сохраняется лучший маршрут. Количество феромона для каждого ребра обновляется: количество увеличивается на ребрах лучшего маршрута, некоторое количество феромона испаряется со всех ребер. «Муравьи» снова распределяются по вершинам, и алгоритм продолжает работу до тех пор, пока не выполнится условие окончания — будет достигнуто максимальное количество итераций или в течение нескольких итераций решение не будет меняться (с заданной точностью).

Метод муравьиной колонии *ACA_TCP* библиотеки Scikit-opt (листинг 2) имеет следующие параметры:

func — целевая функция, длина маршрута,
n_dim — количество городов,
size_pop — размер популяции, количество муравьев,
max_iter — максимальное количество итераций,
distance_matrix — матрица расстояний,
alpha — значимость феромона,
beta — значимость расстояния,
rho — коэффициент испарения феромона.

Листинг 2

Пример применения метода ACA_TCP

```
#билиотека Scikit-opt требует импорта numpy
import numpy as np
from sko.ACA import ACA_TCP
#функция расчета длины маршрута
def path_length(path):
    n_points = path.shape[0]
    return sum([dist_matrix[path[i % n_points],
path[(i + 1) % n_points]] for i in range(n_points)])
# используем dist_matrix (листинг 1)
n_points = len(dist_matrix)
ants = ACA_TSP(func=path_length, n_dim=n_points,
               size_pop=100, max_iter=200,
               distance_matrix=dist_matrix)
best_points, best_distance = ants.run()
```

Генетические алгоритмы основаны на принципах биологической эволюции, имитируют процессы естественного отбора, механизмы наследственности и изменчивости. В генетическом алгоритме начальная популяция хромосом (возможных решений) эволюционирует: на каждой итерации к текущей популяции применяются операторы отбора (селекции), скрещивания, мутации. Хромосомы каждого поколения оцениваются с помощью функции приспособленности (целевой функции задачи). Итерационный процесс продолжается до выполнения одного или нескольких условий останова (стабилизация популяции, максимальное количество итераций, время выполнения и др.).

В ЗК хромосома кодируется последовательностью вершин графа и представляет некоторый маршрут, функция приспособленности задает длину (стоимость) маршрута.

Операторы отбора, скрещивания и мутации носят вероятностный характер. Например, один из методов отбора — знакомое по муравьиному алгоритму «колесо рулетки». Хромосомы отбираются в качестве «родителей» следующего поколения с вероятностью, пропорциональной их приспособленности. Мутация (случайное точечное изменение хромосомы) происходит не в обязательном порядке, а с некоторой вероятностью.

Метод *GA_ TCP*, реализующий генетический алгоритм для ЗК, библиотеки Scikit-opt (листинг 3) имеет следующие параметры:
func — целевая функция, длина маршрута,
n_dim – количество городов,
size_pop — размер популяции, количество хромосом,
max_iter — максимальное количество итераций,
prob_mut — вероятность мутации.

Листинг 3

Пример применения метода GA_TSP

```
import numpy as np
import sko
#функция расчета длины маршрута
def path_length(path):
    n_points = path.shape[0]
    return sum([dist_matrix[path[i % n_points],
path[(i + 1) % n_points]] for i in range(n_points)])
n_points = len(dist_matrix)
chrom = sko.GA.GA_TSP(func=path_length,
    n_dim=n_points, size_pop=200, max_iter=4000, prob_mut=0.3)
best_points, best_distance = chrom.run()
print(best_points, best_distance)
```

3. Результаты и обсуждение

Набор данных bays29.tsp

Длина оптимального маршрута для набора данных bays29.tsp равна 2020. Была проведена серия запусков алгоритма муравьиной колонии при различных сочетаниях параметров. Длина лучшего из найденных маршрутов составила 2030 при значениях параметров: количество муравьев — 29 (число вершин), максимальное число итераций — 150, значимость феромона — 1, значимость расстояния — 5, коэффициент испарения феромона — 0.05. Затем проведена серия запусков алгоритма при

этих значениях параметров. Некоторые полученные маршруты приведены в табл. 1. Отклонение всех полученных решений от оптимального находится в пределах 3 %. Время выполнения программы около 10 секунд. Дальнейшее увеличение количества муравьев и итераций не приводит к улучшению: разброс получаемых маршрутов остается приблизительно таким же.

Таблица 1

Результаты серии запусков алгоритма муравьиной колонии на наборе данных bays29.tsp

	Маршрут	**Длина**
	0 27 5 11 8 4 25 28 2 1 19 9 3 14 17 16 13 21 10 18 24 6 22 26 7 23 15 12 20	2020
1	**0 27 5 11 8 4 25 28 2 1** 20 12 9 19 3 14 17 13 16 21 10 18 24 6 22 26 15 23 7	2030
2	**0 27 5 11 8 4 25 28 2 1** 20 19 9 3 14 17 16 13 21 10 18 12 15 24 6 22 26 7 23	2035
3	**0 27 5 11 8 4 25 28 2 1** 20 19 9 3 14 17 16 13 21 10 18 24 6 22 26 7 23 15 12	2039
4	**0 27 5 11 8 4 25 28 2 1** 20 19 9 3 14 17 13 16 21 10 18 24 6 22 26 7 23 15 12	2041
5	**0 27 5 11 8 4 25 28 2 1** 20 19 9 3 14 17 16 13 21 10 18 24 6 22 26 23 12 15 7	2053
6	**0 27 5 11 8 4 25 28 2 1** 20 19 9 3 14 13 17 16 21 10 18 24 6 22 26 7 23 15 12	2063
7	**0 27 5 11 8 4 25 28 2 1** 19 9 3 14 17 13 21 16 **10 18 24 6 22 26 7 23 15 12 20**	2069
	n_dim=29, size_pop=29, max_iter=150, alpha=1, beta=5, rho=0.05	

Далее была проведена серия запусков генетического алгоритма. Маршруты, длина которых отличается от оптимальной не более чем на 5 %, были получены при количестве хромосом 200, максимальном числе итераций — 2000, вероятности мутации — 0.3. Время работы программы около 1 минуты. Некоторые полученные маршруты приведены в табл. 2. За всю серию запусков программы два раза алгоритм нашел оптимальный маршрут.

Таблица 2

Результаты серии запусков генетического алгоритма на наборе данных bays29.tsp

	Маршрут	**Длина**
	0 27 5 11 8 4 25 28 2 1 19 9 3 14 17 16 13 21 10 18 24 6 22 26 7 23 15 12 20	2020
1	9 3 14 17 16 13 21 10 18 24 6 22 26 7 23 15 12 20 0 27 5 11 8 4 25 28 2 1 19	2020
2	25 4 8 11 5 27 0 20 12 15 23 7 26 22 6 24 18 10 21 13 16 17 14 3 9 19 1 2 28	2020
3	18 10 21 13 16 17 14 3 9 19 20 1 2 28 25 4 8 11 5 27 0 7 23 12 15 26 22 6 24	2026
4	15 18 12 20 0 27 5 11 8 4 25 28 2 1 19 9 3 14 17 16 13 21 10 24 6 22 26 7 23	2045
5	17 14 18 3 9 19 12 20 1 2 28 25 4 8 11 5 27 0 23 15 26 7 22 6 24 10 21 13 16	2072
6	9 3 17 16 13 21 10 14 18 15 24 6 22 26 7 23 12 0 27 5 11 8 4 25 28 2 1 20 19	2097
7	2 1 20 0 12 19 9 3 14 17 16 13 21 10 24 6 18 15 23 26 22 7 27 5 11 8 4 25 28	2116
	n_dim=29, size_pop=200, max_iter=2000, prob_mut=0.3	

Если сравнить результаты работы алгоритма муравьиных колоний и генетического алгоритма, то можно заметить, что маршруты, найденные алгоритмом муравьиных колоний, начинаются с одной и той же последовательности городов — 0 27 5 11 8 4 25 28 2 1. . . ; а маршруты, построенные генетическим алгоритмом, все разные.

Набор данных d198.tsp

Длина оптимального маршрута для набора данных d198.tsp равна 15780. Была проведена серия запусков алгоритма муравьиной колонии при различных сочетаниях параметров, параметры подбирались так, чтобы время выполнения программы составляло около 5 минут. Установлено, что если количество муравьев находится в пределах от 100 до 120, количество итераций в пределах от 200 до 220, значимость феромона равна 1, значимость расстояния — 2, коэффициент испарения феромона — 0.1, то алгоритм находит маршруты, длина которых отклоняется от оптимальной в пределах 15 %. Результаты одной серии испытаний представлены в табл. 3.

Таблица 3

Результаты серии запусков алгоритма муравьиной колонии на наборе данных d198.tsp

Количество итераций	**Длина маршрута**	**Время выполнения, сек**	**Отклонение, %**
100	18852	147.87	19.47
120	18671	179.27	18.32
140	18384	214.00	16.50
160	17872	240.13	13.26
180	17621	268.30	11.67
200	17489	290.20	10.83
220	17444	320.32	10.55
n_dim=198, size_pop=100, alpha=1, beta=2, rho=0.1			

Далее была проведена серия запусков генетического алгоритма при различных сочетаниях параметров, параметры подбирались так, чтобы время выполнения программы составляло около 5 минут. Длина лучшего найденного маршрута равна 16611, отклонение от оптимального маршрута составляет 5.3 %. Если размер популяции уменьшается от 700 до 100, количество итераций соответственно увеличивается от 1124 до 7624, вероятность мутации — 0.3, то алгоритм находит маршруты, длина которых отклоняется от оптимальной в пределах 10 % (табл. 4).

Таблица 4

Результаты серии запусков генетического алгоритма на наборе данных d198.tsp

Размер популяции	**Количество итераций**	**Длина маршрута**	**Время, сек**	**Отклонение, %**
1000	785	17784	292.18	12.70
900	883	17917	297.40	13.54
800	995	17363	295.43	10.00
700	1124	17139	293.13	8.60
600	1293	16846	288.47	6.75
500	1583	16797	295.69	6.44
400	1971	16867	293.66	6.88
300	2682	16895	299.95	7.00
200	3915	16611	290.76	5.26
100	7624	16688	285.05	5.75
n_dim=198, prob_mut=0.3				

Итак, на небольшом наборе данных bays29 генетический и муравьиный алгоритмы показали сравнимые результаты. На большем наборе данных d198 при ограничении на время работы программы в 5 минут генетический алгоритм показал более точные результаты (табл. 5).

Таблица 5

Сравнение лучших маршрутов, полученных при помощи генетического алгоритма и алгоритма муравьиной колонии на наборе данных d198.tsp

Алгоритм	**Время работы, сек**	**Длина маршрута**	**Отклонение, %**
Муравьиный	290.20	17488.67	10.83
Генетический	290.76	16611.00	5.26

Для набора данных bays29.tsp в работе [15] приводятся результаты тестирования генетического алгоритма решения ЗК, программа разработана авторами. К сожалению, авторы не указали, на каком языке написана программа. При размере популяции 100 хромосом, вероятности скрещивания, равной 1, вероятности мутации, равной 1, точное решение задачи было получено на итерации 18907, время оптимизации около 39 секунд. В серии 50 запусков программы были получены маршруты длиной от 2020 до 2056. Результаты сопоставимы с результатами, полученными нами для библиотеки Scikit-opt.

4. Заключение

Студенты направления «Прикладная информатика» СГУ им. Питирима Сорокина изучают генетический и муравьиный алгоритмы в рамках дисциплины «Математические методы в экономике». Студенты направления «Прикладная математика и информатика» реализуют эти алгоритмы в рамках самостоятельной работы при изучении дисциплины «Исследование операций», а также при выполнении курсовых и выпускных квалификационных работ. Тестирование муравьиного и генетического алгоритмов библиотеки Scikit-opt показало по точности и времени работы результаты, позволяющие использовать библиотеку Scikit-opt на лабораторных занятиях, при выполнении курсовых работ. На лабораторных занятиях студенты самостоятельно реализуют алгоритмы на языке Python и могут сравнить временную эффективность своих программ с эффективностью реализации алгоритмов в библиотеке Scikit-opt.

Список источников

1. **Запорожец Д. Ю.** Комбинированный алгоритм решения трансвычислительных задач // *Информатика, вычислительная техника и инженерное образование*. 2018. № 1 (32). С. 10–20.

2. **Кобак В. Г., Поркшеян В. М., Жуковский А. Г., Пешкевич А. А.** Решение задачи коммивояжёра гибридной генетической моделью при использовании путевого подхода // *Известия вузов. Северо-Кавказский регион. Серия: Технические науки*. 2017. № 2 (194). С. 5–9. DOI: 10.17213/0321-2653-2017-2-5-9.

3. **Щербина О. А.** Метаэвристические алгоритмы для задач комбинаторной оптимизации (обзор) // *ТВИМ*. 2014. № 1 (24). С. 56–72.

4. **Яшин С. Н., Яшина Н. И., Кошелев Е. В., Иванов А. А.** Метаэвристические алгоритмы в управлении инновациями : монография. Н. Новгород: ООО «Печатная мастерская "РАДОНЕЖ"», 2023. 200 с.

5. **Пантелеев А. В., Скавинская Д. В., Алешина Е. А.** Метаэвристические алгоритмы поиска оптимального программного управления : учебник. М.: Инфра-М, 2020. 396 с.

6. Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB // *IOP Conference Series Materials Science and Engineering.* May 2020 847:012027 DOI: 10.1088/1757-899X/847/1/012027 URL: https://www.researchgate.net/publication/341711573_Using_the_Bees_Algorithm_to_solve_combinatorial_optimisation_problems_for_TSPLIB (дата обращения: 27.05.2024).

7. Aco-routing documentation. URL: https://pypi.org/project/aco-routing/ (дата обращения: 21.04.2024).

8. ACO documentation. URL: https://pypi.org/project/aco/ (дата обращения: 21.04.2024).

9. Pygad documentation. URL: https://pypi.org/project/pygad/ (дата обращения: 21.04.2024).

10. DEAP documentation. URL: https://pypi.org/project/deap/ (дата обращения: 21.04.2024).

11. Scikit-opt documentation. URL: https://scikit-opt.github.io/scikit-opt//en/README (дата обращения: 21.04.2024).

12. **Reinelt G.** TSPLIB95. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf (дата обращения: 21.04.2024).

13. Discrete and Combinatorial Optimization. URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (дата обращения: 21.04.2024).

14. TSPLIB95 documentation. URL: https://tsplib95.readthedocs.io/en/stable/pages/usage.html (дата обращения: 21.04.2024).

15. **Андреев А. М., Штуца И. М.** Исследование вычислительной (временной) сложности генетического алгоритма на примере решения задачи коммивояжера // *Вестник Ижевского государственного технического университета.* 2008. № 4 (40). С. 144–146.

References

1. **Zaporozhets D. Yu.** Combined algorithm for solving transcomputational problems. *Informatika, vychislitel'naya tekhnika i inzhenernoye obrazovaniye* [Informatics, computing and engineering education]. 2018. No 1 (32). Pp. 10–20. (In Russ.)

2. **Kobak V. G., Porksheyan V. M., Zhukovsky A. G., Peshkevich A. A.** Solving the traveling salesman problem with a hybrid genetic model using the path approach. *Izvestiya vuzov. Severo-Kavkazskiy region. Seriya: Tekhnicheskiye nauki* [News of universities. North Caucasus region. Series: Technical Sciences]. 2017. No 2 (194). Pp. 5–9. DOI: 10.17213/0321-2653-2017-2-5-9. (In Russ.)

3. **Shcherbina O. A.** Metaheuristic algorithms for combinatorial optimization problems (review). *TVIM* [TWIM]. 2014. No 1 (24). Pp. 56–72. (In Russ.)

4. **Yashin S. N., Yashina N. I., Koshelev E. V., Ivanov A. A.** *Metaevristicheskiye algoritmy v upravlenii innovatsiyami : monografiya* [Metaheuristic algorithms in innovation management : monograph]. Nizhniy Novgorod: LLC "Printing Workshop RADONEZH 2023. 200 p. (In Russ.)

5. **Panteleev A. V., Skavinskaya D. V., Aleshina E. A.** *Metaevristicheskiye algoritmy poiska optimal'nogo programmnogo upravleniya : uchebnik* [Metaheuristic algorithms for searching for optimal program control : textbook]. Moscow: Infra-M, 2020. 396 c. (In Russ.)

6. Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB. *IOP Conference Series Materials Science and Engineering.* May 2020 847:012027 DOI:10.1088/1757-899X/847/1/012027 Available at: https://www.researchgate.net/publication/341711573_Using_the_Bees_Algorithm_to_solve_combinatorial_optimisation_problems_for_TSPLIB (accessed: 27.05.2024).

7. *Aco-routing documentation.* Available at: https://pypi.org/project/aco-routing/ (accessed: 21.04.2024).

8. *ACO documentation.* Available at: https://pypi.org/project/aco/ (accessed: 21.04.2024).

9. *Pygad documentation.* Available at: https://pypi.org/project/pygad/ (accessed: 21.04.2024).

10. *DEAP documentation.* Available at: https://pypi.org/project/deap/ (accessed: 21.04.2024).

11. *Scikit-opt documentation.* Available at: https://scikit-opt.github.io/scikit-opt//en/README (accessed: 21.04.2024).

12. **Reinelt G.** *TSPLIB95.* Available at: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf (accessed: 21.04.2024).

13. *Discrete and Combinatorial Optimization.* Available at: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (accessed: 21.04.2024).

14. *TSPLIB95 documentation.* Available at: https://tsplib95.readthedocs.io/en/stable/pages/usage.html (accessed: 21.04.2024).

15. **Andreev A. M., Shtutsa I. M.** Study of the computational (time) complexity of a genetic algorithm using the example of solving the traveling salesman problem. *Vestnik Izhevskogo gosudarstvennogo tekhnicheskogo universiteta* [Bulletin of Izhevsk State Technical University]. 2008. No 4 (40). Pp. 144–146. (In Russ.)

Сведения об авторах / Information about authors
Бабикова Надежда Николаевна / Nadezhda N. Babikova
к.пед.н., доцент, доцент кафедры прикладной информатики / Candidate of Pedagogical Sciences, Associate Professor, Associate Professor of Applied Informatics Department
Сыктывкарский государственный университет имени Питирима Сорокина / Pitirim Sorokin Syktyvkar State University
167001, Россия, г. Сыктывкар, Октябрьский пр., 55 / 55, Oktyabrsky Ave., Syktyvkar, 167001, Russia