

ЛАБОРАТОРНА РОБОТА № 7

ДОСЛІДЖЕННЯ МУРАШИНИХ АЛГОРИТМІВ

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python навчитися дослідити метод мурашиних колоній.

Хід роботи:

Лістинг програми:

```
import random as rn
import numpy as np
from numpy.random import choice as np_choice

class AntAlgorithm(object):
    def __init__(self, distances, start, n_ants, n_best, n_iterations, decay,
alpha=1, beta=1):
        # Ініціалізація класу AntAlgorithm з необхідними параметрами
        self.distances = distances
        self.start = start
        self.pheromone = np.ones(self.distances.shape) / len(distances)
        self.all_inds = range(len(distances))
        self.n_ants = n_ants
        self.n_best = n_best
        self.n_iterations = n_iterations
        self.decay = decay
        self.alpha = alpha
        self.beta = beta
        self.current_position = start
        self.next_position = None
        self.results = []

    def get_started(self):
        shortest_path = None # Найкоротший маршрут на поточній ітерації
        all_time_shortest_path = ("placeholder", np.inf) # Найкоротший
маршрут за всі ітерації
        for i in range(1, self.n_iterations + 1):
            all_paths = self.get_rout() # Отримання всіх маршрутів мурашок
            self.spread_pheromone(all_paths, self.n_best, shortest_path)
            # Розповсюдження феромонів
            shortest_path = min(all_paths, key=lambda x: x[1]) #
Знаходження найкоротшого маршруту на поточній ітерації
            if shortest_path[1] < all_time_shortest_path[1]:
```

					ДУ «Житомирська політехніка».23.122.4.000 – Лр7			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Дяченко В.В.			Звіт з лабораторної роботи	Лім.	Арк.	Аркушів
Перевір.							1	
Керівник						ФІКТ Гр. КН-20-1(1)		
Н. контр.								
Зав. каф.								

```

        all_time_shortest_path = shortest_path # Оновлення
найкоротшого маршруту за всі ітерації
        self.pheromone = self.pheromone * self.decay # Зменшення
феромонів

        # Виведення результату кожної 50-ї ітерації та останньої
ітерації
        if i % 50 == 0 or i == self.n_iterations:
            print("Step {}: {}".format(i, shortest_path[0]))
            print("Total Distance: {}
km".format(all_time_shortest_path[1]))
            self.results.append((i, shortest_path[0],
all_time_shortest_path[1]))

        return all_time_shortest_path, self.results if self.n_iterations >
0 else (None, [])

def spread_pheronome(self, all_paths, n_best, shortest_path):

    # Розповсюдження феромонів на шляхах з найкращими маршрутами.

    # :param all_paths: Список усіх маршрутів
    # :param n_best: Кількість кращих маршрутів
    # :param shortest_path: Найкоротший маршрут на попередній ітерації

    sorted_paths = sorted(all_paths, key=lambda x: x[1])
    for path, dist in sorted_paths[:n_best]:
        for move in path:
            self.pheromone[move] += 1.0 / self.distances[move]

def get_total_distance(self, path):

    # Отримання загальної відстані для даного маршруту.
    # :param path: Маршрут
    # :return: Загальна відстань маршруту

    total_dist = 0
    for ele in path:
        total_dist += self.distances[ele]
    return total_dist

def get_rout(self):

    # Отримання маршрутів для всіх мурашок.
    # :return: Список маршрутів для всіх мурашок

    all_paths = []
    for i in range(self.n ants):

```

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Лр7	Арк.
						2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        path = self.get_path(self.current_position)
        all_paths.append((path, self.get_total_distance(path)))
    return all_paths

def get_path(self, start):

    # Отримання маршруту для одного мурашки.

    # :param start: Початкове місто для мурашки
    # :return: Маршрут для мурашки

    path = []
    tabu = set()
    tabu.add(start)
    previous_step = start
    for i in range(len(self.distances) - 1):
        move = self.get_next_position(self.pheromone[previous_step],
self.distances[previous_step], tabu)
        path.append((previous_step, move))
        previous_step = move
        tabu.add(move)
    self.next_position = start
    path.append((previous_step, start))
    self.current_position = self.next_position
    return path

def get_next_position(self, pheromone, dist, tabu):

    # Отримання наступної позиції для мурашки.

    # :param pheromone: Матриця феромонів для поточної позиції
    # :param dist: Матриця відстаней для поточної позиції
    # :param tabu: Множина заборонених позицій
    # :return: Наступна позиція для мурашки

    pheromone = np.copy(pheromone)
    pheromone[list(tabu)] = 0

    row = pheromone ** self.alpha * ((1.0 / dist) ** self.beta)

    norm_row = row / row.sum()
    move = np_choice(self.all_inds, 1, p=norm_row)[0]
    return move

n_ants = len(table_cities)
n_best = 3
n_iterations = 250
decay = 0.95
start = 3 # variant 4

```

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Лр7	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		3

```

ant_algorithm = AntAlgorithm(table_distances, start, n_ants, n_best,
n_iterations, decay, 1, 1)
final_path, iteration_results = ant_algorithm.get_started()

import matplotlib.pyplot as plt

# Витягуємо індекси з найкоротшого маршруту та збільшуємо кожне значення на 1
path_indices = [i[0] + 1 for i in final_path[0]]
num_cities = 25
# Вказуємо позначки на осі x як числа від 1 до 25
x_ticks = range(1, num_cities + 1)

# Отримання результатів для кожної 50-ї ітерації
for result in iteration_results:
    iteration_number, iteration_path, iteration_distance = result

    # Витягуємо індекси з маршруту ітерації та збільшуємо кожне значення на 1
    iteration_indices = [i[0] + 1 for i in iteration_path]

    # Побудова графіка для маршруту ітерації зі спеціальними позначками та
    відзначеннями на осі y
    fig, ax = plt.subplots(figsize=(12, max(6, 0.3 * num_cities)))
    ax.plot(x_ticks, iteration_indices, marker='o', linestyle='-', col-
or='red', label=f'Iteration {iteration_number} ({iteration_distance:.2f} km)')
    ax.set_yticks(range(1, num_cities + 1))
    ax.set_yticklabels(table_cities, fontsize=10) # Використовуємо назви
міст як мітки на осі y
    ax.set_xticks(x_ticks)
    plt.title(f'Маршрут пройдений мікрояксером (Ітерація {itera-
tion_number})', fontsize=16)
    plt.xlabel('Номера міст', fontsize=14)
    plt.ylabel('Назви міст', fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.legend()
    plt.show()

import numpy as np
table_distances = np.array( [
[ np.inf, 645, 868, 125, 748, 366, 256, 316, 1057, 382, 360, 471,
428, 593, 311, 844, 602, 232, 575, 734, 521, 120, 343, 312, 396],
#1
[645, np.inf, 252, 664, 81, 901, 533, 294, 394, 805, 975, 343, 468,
196, 957, 446, 430, 877, 1130, 213, 376, 765, 324, 891, 672], #2
[868, 252, np.inf, 858, 217, 1171, 727, 520, 148, 1111, 1221, 611,
731, 390, 1045, 591, 706, 1100, 1391, 335, 560, 988, 547, 1141, 867],
#3
[125, 664, 858, np.inf, 738, 431, 131, 407, 1182, 257, 423, 677,
557, 468, 187, 803, 477, 298, 671, 690, 624, 185, 321, 389,
271], ...

```

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Лр7	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

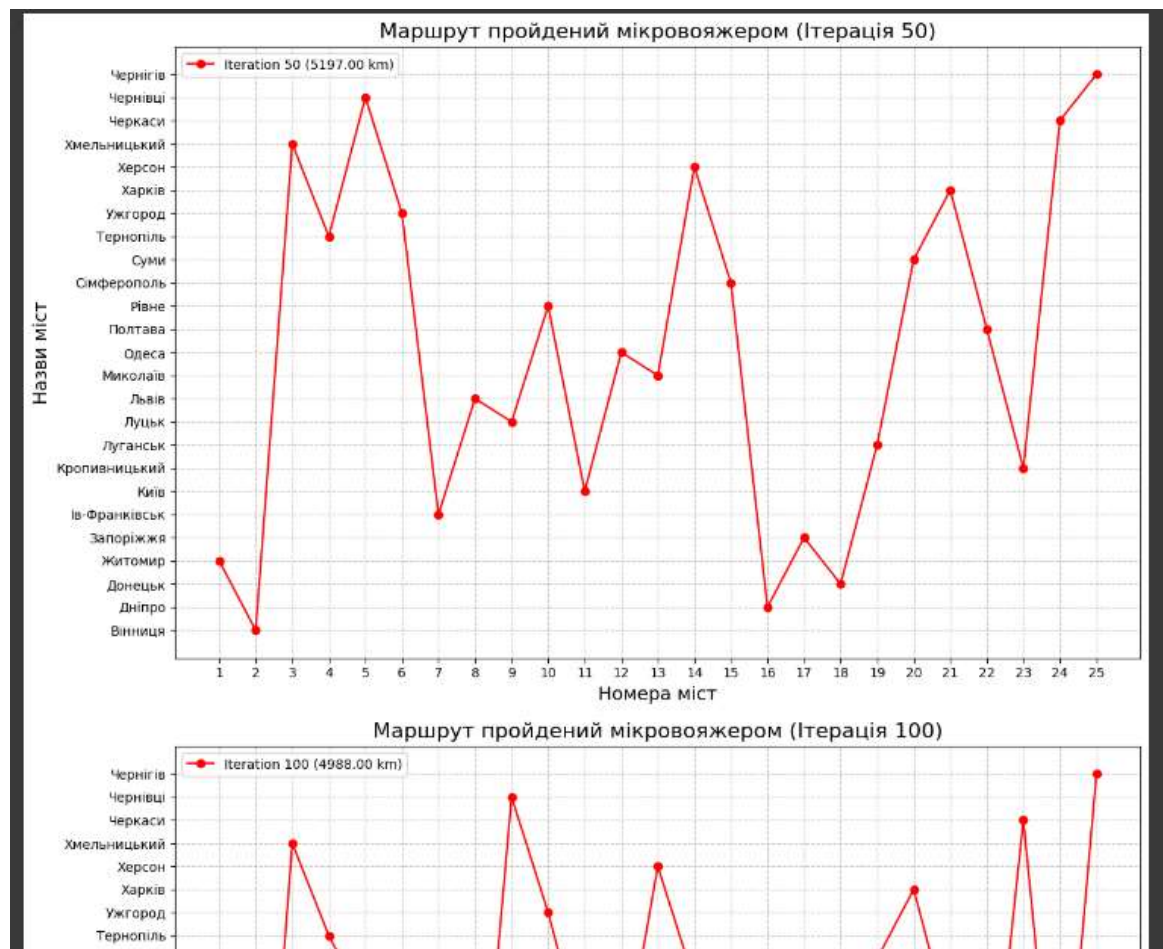


Рис.1. Результат роботи програми.

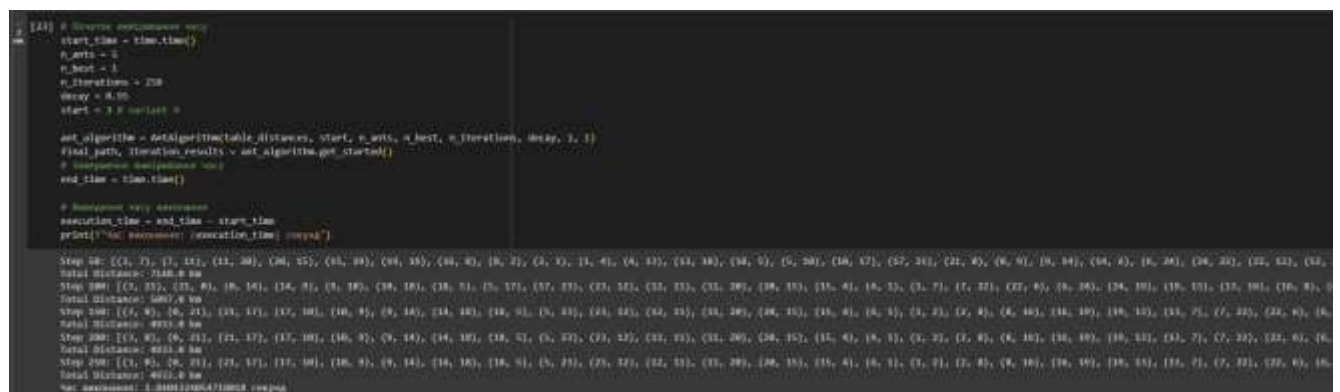


Рис.2. Задача 1.

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Лр7	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

```
# Python implementation map
start_time = time.time()
n_cities = len(table_cities)
n_best = 1
n_iterations = 10
decay = 0.95
start = 0 # variant 1

def algorithm(table_distances, start, n_cities, n_best, n_iterations, decay, i, t):
    final_path, iteration_results = alg_algorithm.get_started()
    # Recursion termination test
    end_time = time.time()

    # Recursion map execution
    execution_time = end_time - start_time
    print("%s execution: (execution_time) output")

Step 10: [(1, 0), (0, 25), (32, 27), (37, 30), (33, 3), (5, 10), (10, 10), (10, 35), (34, 0), (0, 0), (3, 12), (11, 30), (10, 5), (6, 36), (36, 42), (34, 35), (25, 4), (3, 4), (5, 33), (33, 8), (8, 2), (7, 30), (10, 36), (36,
```

Рис.3. Задача 2.

```

# function to compute path
start_time = time.time()
n_sims = len(table_files)
n_best = 10
n_iterations = 500
decay = 0.50
start = 1.0 * n_sims

def algorithm = def algorithm(table_files, start, n_sims, n_best, n_iterations, decay, x, y)
final_path, final_time, results = def algorithm_get_started()
# initialize variables here
end_time = time.time()

# compute path
iteration_time = end_time - start_time
print('iteration_time: %s' % iteration_time)

Step 0: [1, 0], [0, 1], [0, 2], [1, 1], [1, 2], [0, 3], [1, 3], [0, 4], [1, 4], [1, 5], [1, 6], [0, 7], [1, 7], [0, 8], [1, 8], [0, 9], [1, 9], [0, 10], [1, 10], [0, 11], [1, 11], [0, 12], [1, 12], [0, 13], [1, 13], [0, 14], [1, 14], [0, 15], [1, 15], [0, 16], [1, 16], [0, 17], [1, 17], [0, 18], [1, 18], [0, 19], [1, 19], [0, 20], [1, 20], [0, 21], [1, 21], [0, 22], [1, 22], [0, 23], [1, 23], [0, 24], [1, 24], [0, 25], [1, 25], [0, 26], [1, 26], [0, 27], [1, 27], [0, 28], [1, 28], [0, 29], [1, 29], [0, 30], [1, 30], [0, 31], [1, 31], [0, 32], [1, 32], [0, 33], [1, 33], [0, 34], [1, 34], [0, 35], [1, 35], [0, 36], [1, 36], [0, 37], [1, 37], [0, 38], [1, 38], [0, 39], [1, 39], [0, 40], [1, 40], [0, 41], [1, 41], [0, 42], [1, 42], [0, 43], [1, 43], [0, 44], [1, 44], [0, 45], [1, 45], [0, 46], [1, 46], [0, 47], [1, 47], [0, 48], [1, 48], [0, 49], [1, 49], [0, 50], [1, 50], [0, 51], [1, 51], [0, 52], [1, 52], [0, 53], [1, 53], [0, 54], [1, 54], [0, 55], [1, 55], [0, 56], [1, 56], [0, 57], [1, 57], [0, 58], [1, 58], [0, 59], [1, 59], [0, 60], [1, 60], [0, 61], [1, 61], [0, 62], [1, 62], [0, 63], [1, 63], [0, 64], [1, 64], [0, 65], [1, 65], [0, 66], [1, 66], [0, 67], [1, 67], [0, 68], [1, 68], [0, 69], [1, 69], [0, 70], [1, 70], [0, 71], [1, 71], [0, 72], [1, 72], [0, 73], [1, 73], [0, 74], [1, 74], [0, 75], [1, 75], [0, 76], [1, 76], [0, 77], [1, 77], [0, 78], [1, 78], [0, 79], [1, 79], [0, 80], [1, 80], [0, 81], [1, 81], [0, 82], [1, 82], [0, 83], [1, 83], [0, 84], [1, 84], [0, 85], [1, 85], [0, 86], [1, 86], [0, 87], [1, 87], [0, 88], [1, 88], [0, 89], [1, 89], [0, 90], [1, 90], [0, 91], [1, 91], [0, 92], [1, 92], [0, 93], [1, 93], [0, 94], [1, 94], [0, 95], [1, 95], [0, 96], [1, 96], [0, 97], [1, 97], [0, 98], [1, 98], [0, 99], [1, 99], [0, 100], [1, 100], [0, 101], [1, 101], [0, 102], [1, 102], [0, 103], [1, 103], [0, 104], [1, 104], [0, 105], [1, 105], [0, 106], [1, 106], [0, 107], [1, 107], [0, 108], [1, 108], [0, 109], [1, 109], [0, 110], [1, 110], [0, 111], [1, 111], [0, 112], [1, 112], [0, 113], [1, 113], [0, 114], [1, 114], [0, 115], [1, 115], [0, 116], [1, 116], [0, 117], [1, 117], [0, 118], [1, 118], [0, 119], [1, 119], [0, 120], [1, 120], [0, 121], [1, 121], [0, 122], [1, 122], [0, 123], [1, 123], [0, 124], [1, 124], [0, 125], [1, 125], [0, 126], [1, 126], [0, 127], [1, 127], [0, 128], [1, 128], [0, 129], [1, 129], [0, 130], [1, 130], [0, 131], [1, 131], [0, 132], [1, 132], [0, 133], [1, 133], [0, 134], [1, 134], [0, 135], [1, 135], [0, 136], [1, 136], [0, 137], [1, 137], [0, 138], [1, 138], [0, 139], [1, 139], [0, 140], [1, 140], [0, 141], [1, 141], [0, 142], [1, 142], [0, 143], [1, 143], [0, 144], [1, 144], [0, 145], [1, 145], [0, 146], [1, 146], [0, 147], [1, 147], [0, 148], [1, 148], [0, 149], [1, 149], [0, 150], [1, 150], [0, 151], [1, 151], [0, 152], [1, 152], [0, 153], [1, 153], [0, 154], [1, 154], [0, 155], [1, 155], [0, 156], [1, 156], [0, 157], [1, 157], [0, 158], [1, 158], [0, 159], [1, 159], [0, 160], [1, 160], [0, 161], [1, 161], [0, 162], [1, 162], [0, 163], [1, 163], [0, 164], [1, 164], [0, 165], [1, 165], [0, 166], [1, 166], [0, 167], [1, 167], [0, 168], [1, 168], [0, 169], [1, 169], [0, 170], [1, 170], [0, 171], [1, 171], [0, 172], [1, 172], [0, 173], [1, 173], [0, 174], [1, 174], [0, 175], [1, 175], [0, 176], [1, 176], [0, 177], [1, 177], [0, 178], [1, 178], [0, 179], [1, 179], [0, 180], [1, 180], [0, 181], [1, 181], [0, 182], [1, 182], [0, 183], [1, 183], [0, 184], [1, 184], [0, 185], [1, 185], [0, 186], [1, 186], [0, 187], [1, 187], [0, 188], [1, 188], [0, 189], [1, 189], [0, 190], [1, 190], [0, 191], [1, 191], [0, 192], [1, 192], [0, 193], [1, 193], [0, 194], [1, 194], [0, 195], [1, 195], [0, 196], [1, 196], [0, 197], [1, 197], [0, 198], [1, 198], [0, 199], [1, 199], [0, 200], [1, 200], [0, 201], [1, 201], [0, 202], [1, 202], [0, 203], [1, 203], [0, 204], [1, 204], [0, 205], [1, 205], [0, 206], [1, 206], [0, 207], [1, 207], [0, 208], [1, 208], [0, 209], [1, 209], [0, 210], [1, 210], [0, 211], [1, 211], [0, 212], [1, 212], [0, 213], [1, 213], [0, 214], [1, 214], [0, 215], [1, 215], [0, 216], [1, 216], [0, 217], [1, 217], [0, 218], [1, 218], [0, 219], [1, 219], [0, 220], [1, 220], [0, 221], [1, 221], [0, 222], [1, 222], [0, 223], [1, 223], [0, 224], [1, 224], [0, 225], [1, 225], [0, 226], [1, 226], [0, 227], [1, 227], [0, 228], [1, 228], [0, 229], [1, 229], [0, 230], [1, 230], [0, 231], [1, 231], [0, 232], [1, 232], [0, 233], [1, 233], [0, 234], [1, 234], [0, 235], [1, 235], [0, 236], [1, 236], [0, 237], [1, 237], [0, 238], [1, 238], [0, 239], [1, 239], [0, 240], [1, 240], [0, 241], [1, 241], [0, 242], [1, 242], [0, 243], [1, 243], [0, 244], [1, 244], [0, 245], [1, 245], [0, 246], [1, 246], [0, 247], [1, 247], [0, 248], [1, 248], [0, 249], [1, 249], [0, 250], [1, 250], [0, 251], [1, 251], [0, 252], [1, 252], [0, 253], [1, 253], [0, 254], [1, 254], [0, 255], [1, 255], [0, 256], [1, 256], [0, 257], [1, 257], [0, 258], [1, 258], [0, 259], [1, 259], [0, 260], [1, 260], [0, 261], [1, 26
```

Рис.4. Задача 3.

Табл.1. Порівняльний аналіз.

№ Задачі	ants	best	iterations	decay	Довжина маршруту	Час виконання
1	5	1	250	0.95	4933.0 km	1.8 с
2	25	1	50	0.95	6276.0 km	2.5 с
3	25	10	500	0.99	4799.0 km	20.4 с

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Пр7	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис.5. Графік найкоротшого маршруту з початком у місті згідно варіанту (4).

Висновки: виконавши дану лабораторну роботу, ми дослідили метод мурашиних колоній, як узагальнений результат тестування за різних умов привели таблицю 1 та рисунок 5.

Для тестування створеної програми ми виконали задачу в різних умовах з різними початково заданими даними (для того щоб було простіше зробити висновки про її роботу). Виміряли час виконання та порівняли фінальний результат.

- Задача 1: Мала кількість мурашок і середня к-ть ітерацій для оцінки базового виконання.
- Задача 2: Збільшена кількість мурашок для вивчення їх впливу на результат.
- Задача 3: Використання багатьох кращих маршрутів та великої кількості ітерацій для оцінки впливу параметрів.

Збільшення кількості ітерацій може підвищити точність результатів, як видно за результатами задачі 1 та 2. Щоб зменшити використання ресурсів та час – рекомендується збільшувати к-ть ітерацій, а не мурах відповідно. Використання більшої кількості кращих маршрутів (best) може сприяти знаходженню кращих шляхів. Параметр *desau* впливає на швидкість втрати феромонів, і вищі значення можуть сприяти розгортанню феромонів на кращих маршрутах.

Тож враховуючи час виконання та довжину маршруту, задача 3 має найкращі результати хоча і вимагає більше часу для обчислень (20 с – не найгірший результат).

Найкоротший отриманий шлях з міста під індексом 4 (Житомир) за методом мурашиних колоній складає 4744 км.

		Дяченко В.В.			ДУ «Житомирська політехніка».23.122.4.000 – Лр7	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		