**ЛАБОРАТОРНА РОБОТА №7**

**ДОСЛІДЖЕННЯ МУРАШИНИХ АЛГОРИТМІВ**

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

**Хід роботи:**

**GitHub:** https://github.com/ipz201svo/AI

**Завдання 1:** Дослідження мурашиного алгоритму на прикладі рішення задачі комівояжера.

import math

import random

import matplotlib.pyplot as pl

# file with input data

from data import (

    Ukraine\_dis,

    Ukraine\_map,

    manager\_map,

    manager\_dis,

    mom\_map,

    mom\_dis,

    student\_map,

    student\_dis,

)

# variables

variant = 17  # номер варіанта за журналом

alpha = 1.0

beta = 5.0

rho = 0.5  # P

Q = 10

iterations = 10000

mapping = Ukraine\_map  # cities names

object\_count = len(mapping)

distances = Ukraine\_dis  # distance

y = list()  # graphic for best way

class Ant:

    def \_\_init\_\_(self, parent):

        self.parent = parent

        self.position = variant - 1

        self.start = self.position

        self.totalDist = 0.0

        self.tList = [self.position]

        self.myPheromone = []

    def travel(self):

        if len(self.tList) == object\_count:

            self.tList.remove(self.start)

        p\_array = [0 for \_ in range(object\_count)]

        summa = 0

        # formula for counting probability of visiting the following points

        for i in range(object\_count):

            if i not in self.tList:

                summa += (self.parent.pheromones[self.position][i] \*\* alpha) \* (

                    self.parent.visibility[self.position][i] \*\* beta

                )

        for i in range(object\_count):

            if i not in self.tList:

                try:

                    p\_array[i] = (

                        (self.parent.pheromones[self.position][i] \*\* alpha)

                        \* (self.parent.visibility[self.position][i] \*\* beta)

                        / summa

                    )

                except:

                    pass

        revers = list(

            filter(lambda p: p not in self.tList, [i for i in range(object\_count)])

        )  # check place if city been used for a once

        revers.reverse()

        next\_city = revers[0]

        winner\_num = random.random() \* sum(p\_array)  # choosing next point using random

        for i, probability in enumerate(p\_array):

            winner\_num -= probability

            if winner\_num <= 0:

                next\_city = i

                break

        newd = distances[self.position][next\_city]  # writting in a next city

        self.totalDist += newd if newd > 0 and next\_city not in self.tList else math.inf

        self.tList.append(next\_city)

        self.position = next\_city

    def update\_ways(self):  # refreshing pheromone for visiting border

        self.myPheromone = [

            [0.0 for \_ in range(object\_count)] for \_\_ in range(object\_count)

        ]

        for i in range(1, len(self.tList)):

            k = self.tList[i - 1]

            j = self.tList[i]

            self.myPheromone[k][j] = Q / self.totalDist

class Colony:

    smallestCost = math.inf

    optimal\_way = []

    ants = []

    pheromones = None

    visibility = None

    def \_\_init\_\_(self):

        self.pheromones = [

            [1 / (object\_count \* object\_count) for \_ in range(object\_count)]

            for \_\_ in range(object\_count)

        ]  # initial amount of pheromone for the start

        self.visibility = [

            [0 if i == j else 1 / distances[i][j] for i in range(object\_count)]

            for j in range(object\_count)

        ]  # this is an inverse distance

    def do\_main(self):

        self.smallestCost = math.inf

        self.optimal\_way = []

        for t in range(iterations):  # main cycle

            self.reload\_ants()

            self.move\_ants()

            self.update\_ways()

            y.append(self.smallestCost)  # graphics data

        return self.smallestCost, self.optimal\_way

    def move\_ants(self):

        for ant in self.ants:

            for i in range(object\_count):

                ant.travel()

            if ant.totalDist < self.smallestCost:  # determine the optimal path

                self.smallestCost = ant.totalDist

                self.optimal\_way = [ant.tList[-1]] + ant.tList

            ant.update\_ways()

    def update\_ways(self):  # evaporate and add pheromones

        for i, row in enumerate(self.pheromones):

            for j, col in enumerate(row):

                self.pheromones[i][j] \*= rho

                for ant in self.ants:

                    self.pheromones[i][j] += ant.myPheromone[i][j]

    def reload\_ants(self):  # updating agents

        self.ants = [Ant(self) for \_ in range(round(object\_count \* 0.8))]

newLineSymbol = "\n-> "

dist, path = Colony().do\_main()

print(

    f"Оптимальний результат: {dist}, \nШЛЯХ:\n{newLineSymbol.join(mapping[i] for i in path)}"

)

pl.figure()

pl.plot(

    [x for x in range(object\_count + 1)],

    path,

)

for i, txt in enumerate(mapping):

    pl.annotate(txt, (path.index(i), i))

pl.show()

**Зображення, що містить текст, знімок екрана, схема, ряд

Автоматично згенерований опис**

Рис. 1.1 – Результат виконання програми

Зображення, що містить текст, знімок екрана, документ, меню

Автоматично згенерований опис

Рис. 1.2 – Результат виконання програми

Тестування на інших задачах:

Задача 1: Менеджеру необхідно виконати велику кількість завдань по проекту. Потрібно прорахувати найменший шлях, аби встигнути виконати поставлені задачі до кінця робочого дня. Варіант обрано 9 як половина реального варіанту 17, через обмеження в вхідних даних.

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис. 1.3 – Результат виконання програми

Зображення, що містить текст, ряд, знімок екрана, схема

Автоматично згенерований опис

Рис. 1.4 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана, квитанція

Автоматично згенерований опис

Рис. 1.5 – Результат виконання програми

Задача 2: Є список покупок, проте усі предмети продаються в різних магазинах. Знайти найшвидший шлях щоб купити усі речі. Як змінну варіанта було обрано 5.

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис. 1.6 – Результат виконання програми

Зображення, що містить текст, схема, ряд, Графік

Автоматично згенерований опис

Рис. 1.7 – Результат виконання програми

Зображення, що містить текст, квитанція, Шрифт, алгебра

Автоматично згенерований опис

Рис. 1.8 – Результат виконання програми

Задача 3: На зимовій риболовлі зроблено декілька лунок. Рибалці потрібно обійти їх всіх за найкоротший проміжок часу. Як змінну варіанту обрано – 7.

Зображення, що містить текст, знімок екрана, Шрифт, почерк

Автоматично згенерований опис

Рис. 1.9 – Результат виконання програми

Зображення, що містить ряд, схема, Графік, схил

Автоматично згенерований опис

Рис. 1.10 – Результат виконання програми

Зображення, що містить текст, квитанція, Шрифт, алгебра

Автоматично згенерований опис

Рис. 1.11 – Результат виконання програми

**Висновок:** під час виконання лабораторної роботи, використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити метод мурашиних колоній.