**Российский химико-технологический университет**

**имени Д.И.Менделеева**

**Факультет информационных технологий и управления**

**Отдел дополнительной профессиональной подготовки**

Название предмета: \_\_\_Программирование на языке Java\_\_\_\_\_\_

Преподаватели: \_\_Женса А. В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| № Задания | 1 | 2 | Оценка |
| Балл |  |  |  |

(подпись преподавателя)

Программа представляет собой программную реализацию алгоритма муравьиных колоний для поиска оптимального решения задачи коммивояжера.

Программа реализована в программной среде разработки «IntelliJ IDEA Community Edition» при помощи проекта «Maven».

Входными данными для программы является таблица Excel с матрицами времени выпуска продуктов, времени переналадки аппаратов с продукта на продукт и матрицы пути выпуска продукта на аппаратах.

При работе программа спрашивает название файла для загрузки матриц, а также предлагает загрузить или ввести настройки алгоритма.

После загрузки данных программа рассчитывает оптимальную последовательность, для которой общее время выпуска минимально.

Данные расчета программа предлагает сохранить в файл, а также выводит в окно консоли.

Программа состоит из трех пакетов «antOptimization», «factory» и «parser»

В пакете «antOptimization» хранятся следующие классы:

* Menu: реализует основное меню и вызов функций подготовки данных для расчета, сохранения и вывода данных;
* AntAlgorithm: реализует непосредственно работу алгоритма;
* DataOptimization: служит контейнером для результатов работы алгоритма, для последующего вывода в Excel файл;
* ParameterAntOptimization: обеспечивает хранение настроек муравьиного алгоритма;
* AntColony: обеспечивает хранение настроек колоний и их реализацию;
* Ant: обеспечивает хранение параметров муравьёв и их реализацию.

В пакете «factory».хранятся следующие классы:

* FactoryParameter: обеспечивает хранение матриц времени;
* FactoryParameterCreater: обеспечивает создание и наполнение класса FactoryParameter;
* FactoryParameterReader: обеспечивает чтение данных из файла при помощи класса ExcelReader.
* Schedule: рассчитывает расписание работы оборудования используя данные класса FactoryParameter.

В пакете «parser».хранятся следующие классы:

* ExcelReader, ExcelWriter: обеспечивающие чтение и сохранение данных в Excel файл;
* XmlReader, XmlWriter: обеспечивающие чтение и сохранение настроек алгоритма в Xml файл.

Приложение 1

Код программы

package logic.antOptimization;

public class Main {

public static void main(String[] args) throws IOException {

Menu menu = new Menu();

menu.runProgramm();

}

}

package logic.antOptimization;

public class Menu {

public void runProgramm() throws IOException{

AntColony antColony = createAntColony();

ParameterAntOptimization param = setParameter(antColony);

AntAlgorithm antAlgorithm = new AntAlgorithm();

DataOptimization outData = antAlgorithm.algorithm(param, antColony);

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Cохранить результаты? y/n");

if (sayYes(br.readLine())) {

saveToFileExcel(outData, antColony);

}

System.out.println("Cохранить настройки? y/n");

if (sayYes(br.readLine())) {

paramXmlSave(param, antColony);

}

}

private void paramXmlSave(ParameterAntOptimization param, AntColony antColony) throws IOException {

System.out.println("Введите название файла");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String fileName = br.readLine();

XmlWriter xmlWriter = new XmlWriter();

xmlWriter.write(param, antColony, fileName);

}

private AntColony createAntColony() throws IOException {

AntColony ac = new AntColony();

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Добро пожаловать в муравьиный алгоритм");

System.out.println("Загрузить предыдущие настройки колоний?");

System.out.println("Да - Y / Нет - N");

String fileNameForLoad;

if ( sayYes(br.readLine()) ) {

System.out.println("Введите название файла для загрузки настроек");

fileNameForLoad = br.readLine().trim().toLowerCase();

XmlReader xmlReader = new XmlReader();

ac = xmlReader.colonyParse(fileNameForLoad);

}

else {

ac.createAntColony();

}

return ac;

}

private ParameterAntOptimization setParameter(AntColony ac) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String fileNameForLoad;

System.out.println("Загрузить предыдущие настройки алгоритма?");

System.out.println("Да - Y / Нет - N");

if ( sayYes(br.readLine()) ) {

System.out.println("Введите название файла для загрузки настроек");

fileNameForLoad = br.readLine();

XmlReader xmlReader = new XmlReader();

ParameterAntOptimization parameterAntOptimization = xmlReader.parameterParse(fileNameForLoad);

return parameterAntOptimization;

}

else {

return new ParameterAntOptimization(ac);

}

}

private void saveToFileExcel(DataOptimization outData, AntColony ac) throws IOException{

LocalDate localDate = LocalDate.now();

System.out.println("Введите название файла для сохранения результатов");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

StringBuffer sb = new StringBuffer(br.readLine().trim().toLowerCase());

sb.append("\_")

.append(Integer.toString(localDate.getDayOfMonth()))

.append('.')

.append(Integer.toString(localDate.getMonth().getValue()))

.append('.')

.append( Integer.toString(localDate.getYear()));

ExcelWriter ew = new ExcelWriter();

ew.setFileNameForSave(sb.toString());

ew.writeEraLengthWay(outData.getOptimaWayList(), outData.getLengthOptimaWayList());

ew.saveConficDistance(ac);

ew.saveConfig(outData, ac);

// ew.paintLineChart(outData);

}

private Boolean sayYes (String str) {

return str.trim().toLowerCase().equals("y");

}

}

package logic.antOptimization;

public class DataOptimization {

private int countColony;

private int countAntInOneColony;

private int countAnts;

private double lengthOptimaWay;

private int maxCountEra;

private double degreeInfluencePheromone;

private double degreeInfluenceDistance;

private double evaporationPheromone;

private int[] optimaWay;

private ArrayList<Double> lengthOptimaWayList = new ArrayList<>();

private ArrayList<int[]> optimaWayList = new ArrayList<>();

private long timeOptimization;

DataOptimization(ParameterAntOptimization param, AntColony ac) {

this.countColony = ac.getCountColony();

this.countAntInOneColony = ac.getCountAntsInOneColony();

this.countAnts = countColony \* countAntInOneColony;

this.lengthOptimaWay = 0;

this.maxCountEra = param.getMaxCountEra() ;

this.degreeInfluencePheromone = param.getDegreeInfluencePheromone() ;

this.degreeInfluenceDistance = param.getDegreeInfluenceDistance() ;

this.evaporationPheromone = param.getEvaporationPheromone() ;

this.optimaWay = new int[countColony];

}

public ArrayList<Double> getLengthOptimaWayList() {

return lengthOptimaWayList;

}

public ArrayList<int[]> getOptimaWayList() {

return optimaWayList;

}

public void addLengthOptimaWay (Double length) {

this.lengthOptimaWayList.add(length);

}

public void addOptimaWay (int[] way) {

this.optimaWayList.add(way);

}

public int getCountColony() {

return countColony;

}

public int getCountAntInOneColony() {

return countAntInOneColony;

}

public int getCountAnts() {

return countAnts;

}

public double getLengthOptimaWay() {

return lengthOptimaWay;

}

public int getMaxCountEra() {

return maxCountEra;

}

public double getDegreeInfluencePheromone() {

return degreeInfluencePheromone;

}

public double getDegreeInfluenceDistance() {

return degreeInfluenceDistance;

}

public double getEvaporationPheromone() {

return evaporationPheromone;

}

public int[] getOptimaWay() {

return optimaWay;

}

public long getTimeOptimization() {

return timeOptimization;

}

public void setTimeOptimization(long timeOptimization) {

this.timeOptimization = timeOptimization;

}

public void setOptimaWay(int[] optimaWay) {

this.optimaWay = optimaWay;

}

public void setLengthOptimaWay(double lengthOptimaWay) {

this.lengthOptimaWay = lengthOptimaWay;

}

public void setLengthOptimaWayList(ArrayList<Double> lengthOptimaWayList) {

this.lengthOptimaWayList = lengthOptimaWayList;

}

public void setOptimaWayList(ArrayList<int[]> optimaWayList) {

this.optimaWayList = optimaWayList;

}

} package logic.antOptimization;

class AntMath {

static double summArray(double Array[][]) {

double summ = 0;

for (double i[]: Array){

for (double j: i){

summ += j;

}

}

return summ;

}

static double roundTo2Decimal(double x) {

x = Math.round(x \* 1);

return x /= 1;

}

static double minIntValue(int[] Array) {

double min = Array[0];

for (int i=1; i<Array.length; i++) {

if (Array[i] < min) {

min = Array[i];

}

}

return min;

}

}

package logic.antOptimization;

@XmlRootElement(name="parameterAntOptimization")

@XmlAccessorType(XmlAccessType.FIELD)

public class AntColony {

@XmlElement(name = "countColony")

private int countColony;

@XmlElement(name = "countAntsInOneColony")

private int countAntsInOneColony;

@XmlElement(name = "distanceBetweenColony")

private double[][] distanceBetweenColony;

public void createAntColony() throws IOException {

assingValueVariableColonyFromConsole();

setDistansFromFactory();

}

public int getCountColony() {

return countColony;

}

public void setCountColony(int countColony) {

this.countColony = countColony;

}

public int getCountAntsInOneColony() {

return countAntsInOneColony;

}

public void setCountAntsInOneColony(int countAntsInOneColony) {

this.countAntsInOneColony = countAntsInOneColony;

}

public double[][] getDistanceBetweenColony() {

return distanceBetweenColony;

}

public void setDistanceBetweenColony(double[][] distanceBetweenColony) {

this.distanceBetweenColony = distanceBetweenColony;

}

private void assingValueVariableColonyFromConsole() throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Введите количество муравейников");

this.countColony = Integer.parseInt(br.readLine().trim());

System.out.println("Введите количество муравьёв в каждом муравейнике");

countAntsInOneColony = Integer.parseInt(br.readLine().trim());

}

private void setDistanceBetweenColony() throws IOException {

System.out.println("Задать расстоянмя автоматически? Y/N ?");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

switch (br.readLine().trim().toLowerCase()) {

case "y":

{

setAutoDistanceBetweenColony();

break;

}

case "n":

{

setCustomDistanceBetweenColony();

break;

}

}

}

private void setCustomDistanceBetweenColony() throws IOException {

System.out.println("В ручном вводе");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

this.distanceBetweenColony = new double[this.countColony][this.countColony];

for (int i = 0; i < this.countColony; i++) {

for (int j = 0; j < this.countColony; j++) {

if (i!=j) {

{

System.out.println("Введите расстояние между " + (i+1) + "-м и " + (j+1) + "-м муравейниками");

this.distanceBetweenColony[i][j] = Double.parseDouble(br.readLine().trim());

}

}

else {

this.distanceBetweenColony[i][j] = 0;

}

}

}

}

private void setAutoDistanceBetweenColony() {

this.distanceBetweenColony = new double[this.countColony][this.countColony];

for (int i = 0; i < countColony; i++) {

for (int j = 0; j < countColony; j++) {

if (i!=j) {

this.distanceBetweenColony[i][j] = 1 + Math.random() \* 100; // AntMath.roundTo2Decimal(AntMath.randomDouble(99.0));

}

else {

this.distanceBetweenColony[i][j] = 0;

}

}

}

}

private void setDistansFromFactory () throws IOException {

FactoryParameterCreater fpc = new FactoryParameterCreater();

FactoryParameter fp = new FactoryParameter();

fp = fpc.changer(fp);

Schedule schedule = new Schedule();

this.distanceBetweenColony = new double[this.countColony][this.countColony];

for (int i = 0; i < countColony; i++) {

for (int j = 0; j < countColony; j++) {

if (i!=j) {

this.distanceBetweenColony[i][j] = schedule.wayTimeCalculation(fp, i, j);

}

else {

this.distanceBetweenColony[i][j] = 0;

}

}

}

}

}

package logic.antOptimization;

public class AntAlgorithm {

public DataOptimization algorithm(ParameterAntOptimization algoritmParametrs, AntColony colony) throws IOException {

inizializeAnts(algoritmParametrs, colony);

DataOptimization dataOut = new DataOptimization(algoritmParametrs, colony);

runAntAlgorithm(algoritmParametrs, colony, dataOut);

outInConsoleDataAntOptimization(dataOut);

System.gc();

return dataOut;

}

private void inizializeAnts(ParameterAntOptimization antAlgoritmParam, AntColony ac) {

for (int antsInOneColony = 0; antsInOneColony < ac.getCountAntsInOneColony(); antsInOneColony++){

for (int IndexFirstColonyInWay = 0; IndexFirstColonyInWay < ac.getCountColony(); IndexFirstColonyInWay++){

Ant.addAnt(new Ant(IndexFirstColonyInWay, antAlgoritmParam, ac));

}

}

}

private DataOptimization runAntAlgorithm(ParameterAntOptimization parametrs, AntColony ac, DataOptimization inputData) throws IOException {

int indexOptimalAnt = Ant.getIngexMinimalLengthWay(Ant.getAntList());

Double lengthWayCurrentOptima = Ant.getAntList().get(indexOptimalAnt).getLengthWay();

Double lengthWayOptima = lengthWayCurrentOptima;

int[] currentOptimaWay = Ant.getAntList().get(indexOptimalAnt).getAntWay();

int[] optimaWay = Ant.getAntList().get(indexOptimalAnt).getAntWay();

ArrayList<Double> lengthWayList = new ArrayList<>();

ArrayList<int[]> wayList = new ArrayList<>();

lengthWayList.add(lengthWayCurrentOptima);

wayList.add(currentOptimaWay);

StringBuffer sb = new StringBuffer("");

int CurrentEra = 1;

int NotChangeMinWay = 0;

Date date = new Date();

long timeOptimization = date.getTime();

do {

parametrs.changePferomoneOnWay(ac);

parametrs.changeProbabityTransitionInColony();

for (int i = 0; i < Ant.getAntList().size(); Ant.getAntList().get(i++).changeWay(parametrs, ac) );

indexOptimalAnt = Ant.getIngexMinimalLengthWay(Ant.getAntList());

lengthWayCurrentOptima = Ant.getAntList().get(indexOptimalAnt).getLengthWay();

currentOptimaWay = Ant.getAntList().get(indexOptimalAnt).getAntWay();

if (lengthWayOptima > lengthWayCurrentOptima) {

optimaWay = currentOptimaWay;

lengthWayOptima = lengthWayCurrentOptima;

}

wayList.add( currentOptimaWay );

lengthWayList.add( lengthWayCurrentOptima );

if ( isChangeWay(lengthWayList) ) {

NotChangeMinWay++;

}

else {

NotChangeMinWay = 0;

}

sb.delete( 0, sb.length() );

sb.append("Эпоха № ")

.append(CurrentEra)

.append(" ")

.append(lengthWayCurrentOptima)

.append(Arrays.toString(currentOptimaWay));

System.out.println(sb);

} while ((++CurrentEra <= parametrs.getMaxCountEra())&(NotChangeMinWay < 5000));

Date date2 = new Date();

timeOptimization -= date2.getTime();

System.out.println("Алгоритм работал: " + (timeOptimization \* (-1)) + " мс");

System.out.println("Эпох пройдено : " + --CurrentEra);

if (5000 == NotChangeMinWay){

System.out.println("На протяжении 5000 эпох путь не улучшался, алгоритм закончен");

}

else{

System.out.println("Домтигнут лимит эпох, алгоритм закончен");

}

inputData.setOptimaWay(optimaWay);

inputData.setLengthOptimaWay(lengthWayOptima);

inputData.setTimeOptimization(timeOptimization);

return inputData;

}

private static Boolean isChangeWay(ArrayList<Double> list) {

if ( 0 == Double.compare( list.get(list.size()-2), list.get(list.size()-1) ) ){

return true;

}

else {

return false;

}

}

private static void outInConsoleDataAntOptimization(DataOptimization dataOut) throws IOException {

System.out.println("Вывод");

System.out.println("Общее количество муравьев:");

System.out.println(Ant.getAntList().size());

System.out.println();

System.out.println("Лучший путь за все эпохи: ");

System.out.println(Arrays.toString(dataOut.getOptimaWay()));

System.out.print("Длина оптимального пути за все эпохи: ");

System.out.println(dataOut.getLengthOptimaWay());

}

}

package logic.antOptimization;

public class Ant {

private static ArrayList<Ant> antList = new ArrayList<>();

private int[] antWay; //Set

private double lengthWay;

Ant(ParameterAntOptimization param, AntColony ac) {

new Ant( (int) (Math.random() \* (ac.getCountColony() + 1)), param, ac );

}

Ant(int firstColony, ParameterAntOptimization param, AntColony ac) {

antWay = new int[ac.getCountColony()];

for(int j = 0; j< antWay.length; antWay[j] = j++) ;

swapValueInArrayWay(antWay.length-1, firstColony);

changeWay(param, ac);

changeLengthWay(ac);

}

public static void addAnt (Ant ant) {

antList.add(ant);

}

public static ArrayList<Ant> getAntList() {

return antList;

}

public int[] getAntWay() {

return antWay;

}

public double getLengthWay() {

return lengthWay;

}

public void changeWay(ParameterAntOptimization algParam, AntColony ac) {

double probabilityTransitionAnt;

int indexNextColony;

boolean condition;

double summProbabiliry;

swapValueInArrayWay(0, antWay.length-1);

int i = 1;

do {

double summArrayProbabilityTransitionAnt = 0;

for (int j = i; j < antWay.length; j++){

summArrayProbabilityTransitionAnt += algParam.getProbabilitiTransitionInColony()[antWay[i-1]][antWay[j]];

}

probabilityTransitionAnt = Math.random() \* summArrayProbabilityTransitionAnt;

condition = false;

summProbabiliry = 0;

indexNextColony = i-1;

do {

summProbabiliry += algParam.getProbabilitiTransitionInColony()[antWay[i-1]][antWay[++indexNextColony]];

if ((probabilityTransitionAnt < summProbabiliry)|(indexNextColony == antWay.length-1)) {

condition = true;

}

} while (!condition);

swapValueInArrayWay(i, indexNextColony);

} while (++i < antWay.length-1);

changeLengthWay(ac);

}

private void swapValueInArrayWay(int firstIndex, int secondIndex) {

int antWayBuffer = antWay[firstIndex];

antWay[firstIndex] = antWay[secondIndex];

antWay[secondIndex] = antWayBuffer;

}

private void changeLengthWay(AntColony ac){

lengthWay = 0;

for (int i = 1; i < antWay.length; i++) {

lengthWay += ac.getDistanceBetweenColony()[antWay[i-1]][antWay[i]];

}

lengthWay += 0;

}

public static int getIngexMinimalLengthWay(ArrayList<Ant> ants) {

double min = ants.get(0).getLengthWay();

int minIndex = 0;

for (Ant ant : ants){

if (ant.getLengthWay() < min) {

min = ant.getLengthWay();

minIndex = ants.indexOf(ant);

}

}

return minIndex;

}

}

package logic.antOptimization;

@XmlRootElement(name="parameterAntOptimization")

@XmlAccessorType(XmlAccessType.FIELD)

public class ParameterAntOptimization {

@XmlElement(name = "averangDistant")

private double averangDistant;

@XmlElement(name = "degreeInfluencePheromone")

private double degreeInfluencePheromone;

@XmlElement(name = "degreeInfluenceDistance")

private double degreeInfluenceDistance;

@XmlElement(name = "evaporationPheromone")

private double evaporationPheromone;

@XmlElement(name = "arrayVisibilityColony")

private double[][] arrayVisibilityColony;

@XmlElement(name = "arrayAmountPheromoneOnWay")

private double[][] arrayAmountPheromoneOnWay;

@XmlElement(name = "probabilitiTransitionInColony")

private double[][] probabilitiTransitionInColony;

@XmlElement(name = "maxCountEra")

private int maxCountEra;

ParameterAntOptimization(AntColony ac) throws IOException {

assingValueVariableFromConsole();

assingLengthArray(ac);

averangDistant = (2 \* AntMath.summArray(ac.getDistanceBetweenColony())) / (ac.getCountColony() \* (ac.getCountColony() - 1));

firstCalculationArray(ac);

changeProbabityTransitionInColony();

}

public double getAverangDistant() {

return averangDistant;

}

public void setAverangDistant(double averangDistant) {

this.averangDistant = averangDistant;

}

public double getDegreeInfluencePheromone() {

return degreeInfluencePheromone;

}

public void setDegreeInfluencePheromone(double degreeInfluencePheromone) {

this.degreeInfluencePheromone = degreeInfluencePheromone;

}

public double getDegreeInfluenceDistance() {

return degreeInfluenceDistance;

}

public void setDegreeInfluenceDistance(double degreeInfluenceDistance) {

this.degreeInfluenceDistance = degreeInfluenceDistance;

}

public double getEvaporationPheromone() {

return evaporationPheromone;

}

public void setEvaporationPheromone(double evaporationPheromone) {

this.evaporationPheromone = evaporationPheromone;

}

public double[][] getArrayVisibilityColony() {

return arrayVisibilityColony;

}

public void setArrayVisibilityColony(double[][] arrayVisibilityColony) {

this.arrayVisibilityColony = arrayVisibilityColony;

}

public double[][] getArrayAmountPheromoneOnWay() {

return arrayAmountPheromoneOnWay;

}

public void setArrayAmountPheromoneOnWay(double[][] arrayAmountPheromoneOnWay) {

this.arrayAmountPheromoneOnWay = arrayAmountPheromoneOnWay;

}

public double[][] getProbabilitiTransitionInColony() {

return probabilitiTransitionInColony;

}

public void setProbabilitiTransitionInColony(double[][] probabilitiTransitionInColony) {

this.probabilitiTransitionInColony = probabilitiTransitionInColony;

}

public int getMaxCountEra() {

return maxCountEra;

}

public void setMaxCountEra(int maxCountEra) {

this.maxCountEra = maxCountEra;

}

private void assingLengthArray(AntColony ac) {

this.arrayVisibilityColony = new double[ac.getDistanceBetweenColony().length][];

this.arrayAmountPheromoneOnWay = new double[ac.getDistanceBetweenColony().length][];

this.probabilitiTransitionInColony = new double[ac.getDistanceBetweenColony().length][];

for (int i = 0; i < ac.getDistanceBetweenColony().length; i++) {

this.arrayVisibilityColony[i] = new double[ac.getDistanceBetweenColony()[i].length];

this.arrayAmountPheromoneOnWay[i] = new double[ac.getDistanceBetweenColony()[i].length];

this.probabilitiTransitionInColony[i] = new double[ac.getDistanceBetweenColony()[i].length];

}

}

private void firstCalculationArray(AntColony ac){

for(int i = 0; i < ac.getDistanceBetweenColony().length; i++) {

for (int j = 0; j < ac.getDistanceBetweenColony()[i].length; j++) {

if (i!=j) {

this.arrayVisibilityColony[i][j] = averangDistant / ac.getDistanceBetweenColony()[i][j];

this.arrayAmountPheromoneOnWay[i][j] = 1;

}

else {

this.arrayVisibilityColony[i][j] = 0;

this.arrayAmountPheromoneOnWay[i][j] = 0;

}

}

}

}

private void assingValueVariableFromConsole() throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Введите коэфициент влияния феромона: ");

this.degreeInfluencePheromone = Double.parseDouble(br.readLine().trim());

System.out.println("Введите коэфициент влияния расстояния между муравейниками: ");

this.degreeInfluenceDistance = Double.parseDouble(br.readLine().trim());

System.out.println("Введите коэфициент учитывающий испарение феромона: ");

this.evaporationPheromone = Double.parseDouble(br.readLine().trim());

System.out.println("Ведите количество эпох работы алгоритма");

this.maxCountEra = Integer.parseInt(br.readLine().trim());

}

public void changePferomoneOnWay(AntColony ac){

for(int i = 0; i < this.arrayAmountPheromoneOnWay.length; i++) {

for (int j = 0; j < this.arrayAmountPheromoneOnWay[i].length; j++) {

if (i!=j) {

this.arrayAmountPheromoneOnWay[i][j] =

(this.evaporationPheromone \* this.arrayAmountPheromoneOnWay[i][j] + this.averangDistant \*

(ac.getCountColony()-1) \* (1 - this.evaporationPheromone) \* calculationSummReciprocalLengthWayGivenColony(i, j));

}

}

}

}

public void changeProbabityTransitionInColony() {

for (int i = 0; i < this.probabilitiTransitionInColony.length; i++) {

for (int j = 0; j < this.probabilitiTransitionInColony[i].length; j++) {

this.probabilitiTransitionInColony[i][j] = (Math.pow(this.arrayAmountPheromoneOnWay[i][j], this.degreeInfluencePheromone)

\* Math.pow(this.arrayVisibilityColony[i][j], this.degreeInfluenceDistance));

}

}

}

private double calculationSummReciprocalLengthWayGivenColony(int indexFirstColony, int indexSecondColony) {

double summReciprocalLength = 0;

for (Ant ant : Ant.getAntList()) {

if (checkSequenceValuesInArray(ant.getAntWay(), indexFirstColony, indexSecondColony)) {

summReciprocalLength += 1 / ant.getLengthWay();

}

}

return summReciprocalLength;

}

private boolean checkSequenceValuesInArray(int[] Array, int firstValue, int secondValue) {

boolean Result = false;

int i = 0;

do {

if((Array[i] == firstValue)&(Array[i+1] == secondValue)) {

Result = true;

}

} while ((!Result)&(++i < Array.length-1));

return Result;

}

}

package logic.factory;

public class Schedule {

private double[] beginTimeWorkApparatsArray;

private double[] endTimeWorkApparatsArray;

private double[][] productTimeArray;

public double calculationAllTime(FactoryParameter inFactory) {

getShedule(inFactory);

double time = productTimeArray[0][1];

for (int i = 1; i < productTimeArray.length; i++){

if (time < productTimeArray[i][1]) {

time = productTimeArray[i][1];

}

}

return time;

}

public double wayTimeCalculation (FactoryParameter inFactory, int firstProduct, int secondProduct){

double time = 0;

int[] seqProduct = {firstProduct, secondProduct};

inFactory.setSequenceProduct(seqProduct);

getShedule(inFactory);

time = Math.abs(productTimeArray[1][0] - productTimeArray[0][1]);

return time;

}

private void getShedule(FactoryParameter factory) {

initializationArrays(factory);

int productIndex = 0;

int nowProductIndex = factory.getSequenceProduct()[productIndex];

int lastProductIndex = 0;

int nowApparatIndex = 0;

int lastApparatIndex = 0;

boolean isFirstApparat = true;

for (int apparat = 0; apparat < factory.apparatCount; apparat++){

// TODO: 18/12/17 say yes

if (0 != (factory.getApparatScheduleArray()[nowProductIndex][apparat]) ){

if (isFirstApparat) {

nowApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], apparat);

endTimeWorkApparatsArray[nowApparatIndex] += factory.getProductTimeReleaseArray()[nowProductIndex][nowApparatIndex];

productTimeArray[productIndex][0] = 0;

isFirstApparat = false;

}

else {

nowApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], apparat);

lastApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], (apparat-1));

beginTimeWorkApparatsArray[nowApparatIndex] = endTimeWorkApparatsArray[lastApparatIndex];

endTimeWorkApparatsArray[nowApparatIndex] = beginTimeWorkApparatsArray[nowApparatIndex] + factory.getProductTimeReleaseArray()[nowProductIndex][nowApparatIndex];

}

}

}

productTimeArray[productIndex][1] = endTimeWorkApparatsArray[nowApparatIndex];

System.out.println("end product 0");

double tempBeginTime = 0;

productIndex++;

for (; productIndex < factory.getSequenceProduct().length; productIndex++) {

nowProductIndex = factory.getSequenceProduct()[productIndex];

lastProductIndex = factory.getSequenceProduct()[productIndex - 1];

isFirstApparat = true;

for (int apparat = 0; apparat < factory.apparatCount; apparat++) {

if (0 != (factory.getApparatScheduleArray()[nowProductIndex][apparat])) {

if (isFirstApparat) {

nowApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], apparat);

beginTimeWorkApparatsArray[nowApparatIndex] = endTimeWorkApparatsArray[nowApparatIndex] +

factory.getChangeoverListArray().get(nowApparatIndex)[lastProductIndex][nowProductIndex];

endTimeWorkApparatsArray[nowApparatIndex] = beginTimeWorkApparatsArray[nowApparatIndex] +

factory.getProductTimeReleaseArray()[nowProductIndex][nowApparatIndex];

productTimeArray[productIndex][0] = beginTimeWorkApparatsArray[nowApparatIndex];

isFirstApparat = false;

} else {

nowApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], apparat);

lastApparatIndex = getIndex(factory.getApparatScheduleArray()[nowProductIndex], (apparat - 1));

tempBeginTime = endTimeWorkApparatsArray[nowApparatIndex] +

factory.getChangeoverListArray().get(nowApparatIndex)[lastProductIndex][nowProductIndex];

if (tempBeginTime > endTimeWorkApparatsArray[lastApparatIndex]) {

beginTimeWorkApparatsArray[nowApparatIndex] = tempBeginTime;

} else {

beginTimeWorkApparatsArray[nowApparatIndex] = endTimeWorkApparatsArray[lastApparatIndex];

}

endTimeWorkApparatsArray[nowApparatIndex] = beginTimeWorkApparatsArray[nowApparatIndex] +

factory.getProductTimeReleaseArray()[nowProductIndex][nowApparatIndex];

}

}

}//apparat

System.out.println("end product " + nowProductIndex);

productTimeArray[productIndex][1] = endTimeWorkApparatsArray[nowApparatIndex];

}//productIndex

return;

}

private int getIndex(int[] array, int value) {

int index = 0;

for (int element : array) {

if (value == element) {

return index;

}

index++;

}

return (index-1);

}

private void initializationArrays(FactoryParameter fp) {

int productCount = fp.getSequenceProduct().length;

this.beginTimeWorkApparatsArray = new double[fp.apparatCount];

this.endTimeWorkApparatsArray = new double[fp.apparatCount];

for (int i = 0; i < fp.apparatCount; i++) {

this.beginTimeWorkApparatsArray[i] = 0;

this.endTimeWorkApparatsArray[i] = 0;

}

this.productTimeArray = new double[productCount][];

for (int i = 0; i < productCount; i++){

this.productTimeArray[i] = new double[2];

}

}

}

package logic.factory;

public class FactoryParameter {

private ArrayList<double[][]> changeoverListArray = new ArrayList<>();

private double[][] productTimeReleaseArray;

private int[][] apparatScheduleArray;

private int[] sequenceProduct;

int productCount;

int apparatCount;

public ArrayList<double[][]> getChangeoverListArray() {

return changeoverListArray;

}

public void setChangeoverListArray(ArrayList<double[][]> changeoverListArray) {

this.changeoverListArray = changeoverListArray;

}

public int[][] getApparatScheduleArray() {

return apparatScheduleArray;

}

public void setApparatScheduleArray(int[][] apparatScheduleArray) {

this.apparatScheduleArray = apparatScheduleArray;

}

public int getProductCount() {

return productCount;

}

public void setProductCount(int productCount) {

this.productCount = productCount;

}

public int getApparatCount() {

return apparatCount;

}

public void setApparatCount(int apparatCount) {

this.apparatCount = apparatCount;

}

public void addChangeoverArray (double[][] changeoverArray) {

this.changeoverListArray.add(changeoverArray);

}

public double[][] getProductTimeReleaseArray() {

return productTimeReleaseArray;

}

public void setProductTimeReleaseArray(double[][] productTimeReleaseArray) {

this.productTimeReleaseArray = productTimeReleaseArray;

}

public int[] getSequenceProduct() {

return sequenceProduct;

}

public void setSequenceProduct(int[] sequenceProduct) {

this.sequenceProduct = sequenceProduct;

}

}

package logic.factory;

public class FactoryParameterCreater {

public FactoryParameter changer (FactoryParameter factoryParameter) throws IOException{

String fileNameForLoad;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Введите количество продуктов");

int countProduct = Integer.parseInt(trimmer(br.readLine()));

factoryParameter.setProductCount( countProduct );

System.out.println("Введите количество аппаратов");

int countApparat = Integer.parseInt(trimmer(br.readLine()));

factoryParameter.setApparatCount( countApparat );

System.out.println("Введите название файла для загрузки");

fileNameForLoad = trimmer(br.readLine()) + ".xls";

FactoryParameterReader fpR = new FactoryParameterReader();

factoryParameter.setChangeoverListArray(fpR.readChangeoverListArray(fileNameForLoad, countApparat));

factoryParameter.setProductTimeReleaseArray(fpR.readProductTimeReleaseArray(fileNameForLoad, countApparat, countProduct));

factoryParameter.setApparatScheduleArray(fpR.apparatScheduleArray(fileNameForLoad));

return factoryParameter;

}

public void fileCreater (String fileName, int countApparat, int countProduct) throws IOException {

ExcelWriter ew = new ExcelWriter();

ew.setFileNameForSave(fileName);

ew.createBook("время выпуска");

ew.createBook("маршрут выпуска");

for (int j = 0; j < countApparat; j++){

ew.createBook("аппарат\_" + j);

}

}

private String trimmer (String text) {

if (!text.isEmpty()) {

text.trim().toLowerCase();

return text;

}

else {

System.out.println("Значение отсутствует");

}

return "1";

}

}

package logic.factory;

public class FactoryParameterReader {

public ArrayList<double[][]> readChangeoverListArray(String fileName, int countApparat) throws IOException{

ArrayList<double[][]> changeoverList = new ArrayList<>();

ExcelReader er = new ExcelReader();

er.setFileNameForLoad(fileName);

String bookNameForLoad = "";

for (int apparat = 0; apparat < countApparat; apparat++){

bookNameForLoad = "аппарат\_" + apparat;

changeoverList.add(er.readDoubleArray(bookNameForLoad));

}

return changeoverList;

}

public double[][] readProductTimeReleaseArray (String fileName, int countApparat, int countProduct) throws IOException{

double[][] result;

ExcelReader er = new ExcelReader();

er.setFileNameForLoad(fileName);

String bookNameForLoad = "время выпуска";

result = er.readDoubleArray(bookNameForLoad);

return result;

}

public int[][] apparatScheduleArray (String fileName) throws IOException{

int[][] result;

int countApparat = 0;

int countProduct = 0;

ExcelReader er = new ExcelReader();

er.setFileNameForLoad(fileName);

String bookNameForLoad = "маршрут выпуска";

double[][] tempArray = er.readDoubleArray(bookNameForLoad);

countProduct = tempArray.length;

countApparat = tempArray[0].length;

result = new int[countProduct][];

for (int i = 0; i < countProduct; i++){

result[i] = new int[countApparat];

for (int j = 0; j < countApparat; j++){

result[i][j] = (int) tempArray[i][j];

}

}

return result;

}

}

package logic.parser;

public class ExcelWriter {

private String fileNameForSave;

public void setFileNameForSave(String fileNameForSave) {

this.fileNameForSave = fileNameForSave + ".xls";

}

public void writeEraLengthWay(ArrayList<int[]> listWay, ArrayList<Double> listLength) throws IOException {

createBook("DataOutOptimization");

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.getSheet("DataOutOptimization");

for (int i = 0; i < listLength.size(); i++) {

Row row = sheet.createRow(i);

Cell Era = row.createCell(0);

Era.setCellValue(i + 1);

Era = row.createCell(1);

Era.setCellValue(listLength.get(i));

Era = row.createCell(2);

Era.setCellValue("Путь: ");

for (int j = 0; j < listWay.get(i).length; j++) {

Era = row.createCell(3 + j);

Era.setCellValue(listWay.get(i)[j]);

}

sheet.autoSizeColumn(i);

}

book.write(new FileOutputStream(fileNameForSave));

book.close();

}

private void writeArray(double[][] Array, String bookName) throws IOException {

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.getSheet(bookName);

Row row;

for (int i = 0; i < Array.length; i++) {

row = sheet.createRow(i);

for (int j = 0; j < Array[i].length; j++) {

Cell Era = row.createCell(j);

Era.setCellValue(Array[i][j]);

sheet.autoSizeColumn(j);

}

}

book.write(new FileOutputStream(fileNameForSave));

book.close();

}

public void saveConficDistance(AntColony ac) throws IOException {

createBook("Матрица путей между колониями");

writeArray(ac.getDistanceBetweenColony(), "Матрица путей между колониями");

}

private void createFile() throws IOException {

Workbook book = new HSSFWorkbook();

book.write(new FileOutputStream(this.fileNameForSave));

book.close();

}

public void createBook(String bookName) throws IOException {

try {

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.createSheet(bookName);

book.write(new FileOutputStream(this.fileNameForSave));

book.close();

} catch (FileNotFoundException e) {

createFile();

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.createSheet(bookName);

book.write(new FileOutputStream(this.fileNameForSave));

book.close();

}

}

public void saveConfig(DataOptimization outData, AntColony ac) throws IOException {

createBook("Параметры алгоритма");

getNameValueParameters("Count colony", ac.getCountColony());

getNameValueParameters("Count Ants In One colony", ac.getCountAntsInOneColony());

getNameValueParameters("Degree Influence Distance", outData.getDegreeInfluenceDistance());

getNameValueParameters("Degree Influence Pheromone", outData.getDegreeInfluencePheromone());

getNameValueParameters("Evaporation Pheromone", outData.getEvaporationPheromone());

getNameValueParameters("Count Era", outData.getMaxCountEra());

getNameValueParameters("Time", outData.getTimeOptimization());

}

private void getNameValueParameters(String cellname, double value) throws IOException {

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.getSheet("Параметры алгоритма");

Row row = sheet.createRow(sheet.getLastRowNum() + 1);

Cell NameParametr = row.createCell(0);

Cell ValueParametr = row.createCell(1);

NameParametr.setCellValue(cellname);

ValueParametr.setCellValue(value);

sheet.autoSizeColumn(0);

book.write(new FileOutputStream(this.fileNameForSave));

book.close();

}

private void getNameValueParameters(String name, int value) throws IOException {

Workbook book = new HSSFWorkbook(new FileInputStream(this.fileNameForSave));

Sheet sheet = book.getSheet("Параметры алгоритма");

Row row = sheet.createRow(sheet.getLastRowNum() + 1);

Cell NameParametr = row.createCell(0);

Cell ValueParametr = row.createCell(1);

NameParametr.setCellValue(name);

ValueParametr.setCellValue(value);

sheet.autoSizeColumn(0);

book.write(new FileOutputStream(this.fileNameForSave));

book.close();

}

package logic.parser;

public class ExcelReader {

private String fileNameForLoad;

public double readValue(String bookName, int i, int j) throws IOException {

double result = 0;

HSSFWorkbook myExcelBook = new HSSFWorkbook(new FileInputStream(fileNameForLoad));

HSSFSheet myExcelSheet = myExcelBook.getSheet(bookName);

HSSFRow row = myExcelSheet.getRow(i);

{

result = row.getCell(j).getNumericCellValue();

}

myExcelBook.close();

return result;

}

public double[][] readDoubleArray(String bookName) throws IOException {

HSSFWorkbook myExcelBook = new HSSFWorkbook(new FileInputStream(fileNameForLoad));

HSSFSheet myExcelSheet = myExcelBook.getSheet(bookName);

HSSFRow row = myExcelSheet.getRow(0);

int lengthI = myExcelSheet.getLastRowNum()+1;

int lengthJ = row.getLastCellNum();

double[][] result = new double[lengthI][];

for (int i = 0; i < lengthI ; i++) {

result[i] = new double[lengthJ];

row = myExcelSheet.getRow(i);

for (int j = 0; j < lengthJ; j++){

result[i][j] = row.getCell(j).getNumericCellValue();

}

}

return result;

}

public String getFileNameForLoad() {

return fileNameForLoad;

}

public void setFileNameForLoad(String fileNameForLoad) {

this.fileNameForLoad = fileNameForLoad;

}

}

package logic.parser;

public class XmlReader {

public ParameterAntOptimization parameterParse(String fileName) {

try {

fileName = fileName.trim().toLowerCase() + "\_parameter.xml";

File file = new File(fileName);

JAXBContext jaxbContext = JAXBContext.newInstance(ParameterAntOptimization.class);

Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

return (ParameterAntOptimization) jaxbUnmarshaller.unmarshal(file);

} catch (JAXBException e) {

e.printStackTrace();

}

return null;

}

public AntColony colonyParse(String fileName) {

try {

fileName = fileName.trim().toLowerCase() + "\_colony.xml";

File file = new File(fileName);

JAXBContext jaxbContext = JAXBContext.newInstance(AntColony.class);

Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();

return (AntColony) jaxbUnmarshaller.unmarshal(file);

} catch (JAXBException e) {

e.printStackTrace();

}

return null;

}

}

package logic.parser;

public class XmlWriter {

public void write (ParameterAntOptimization param, AntColony colony, String fileName) {

try {

fileName = fileName.trim().toLowerCase();

File file = new File(fileName + "\_parameter.xml");

JAXBContext jaxbContext= JAXBContext.newInstance(ParameterAntOptimization.class);

Marshaller marshaller = jaxbContext.createMarshaller();

// output pretty printed

marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);

marshaller.marshal(param, file);

file = new File(fileName + "\_colony.xml");

jaxbContext = JAXBContext.newInstance(AntColony.class);

marshaller = jaxbContext.createMarshaller();

// output pretty printed

marshaller.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, true);

marshaller.marshal(colony, file);

}

catch (JAXBException e) {

e.printStackTrace();

}

}

}