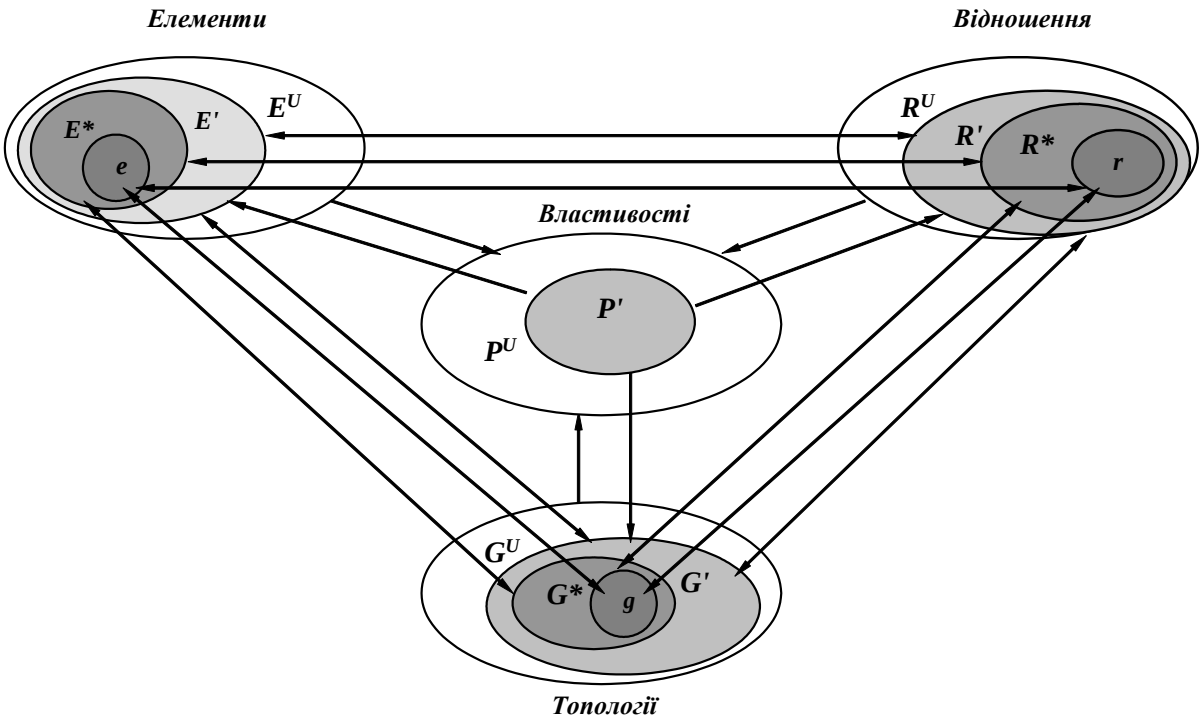
Додаток А

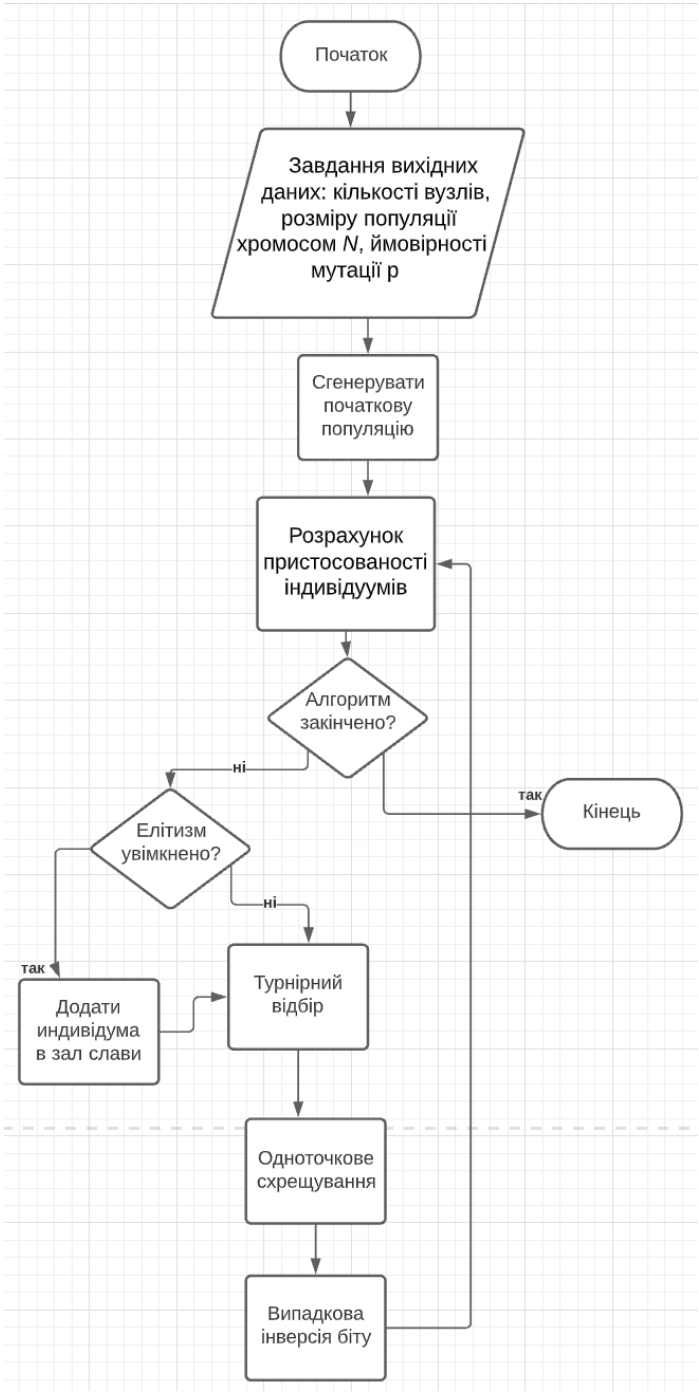
Графічний матеріал атестаційної роботи

Схема зв'язку категорій "елементи", "відношення", "топологія" та "властивості"



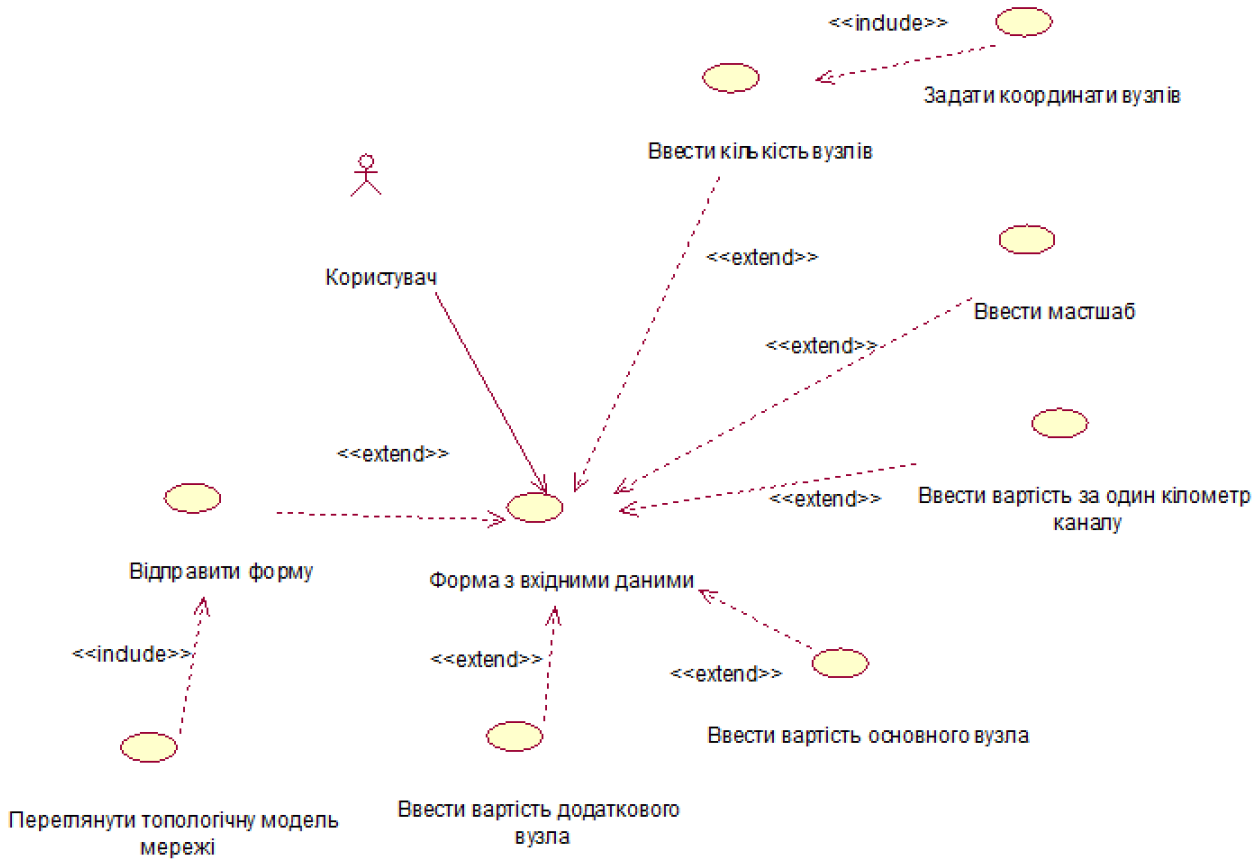
Розробив	Лясковка Я.І.		14.12.21	Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж	
Перевірив	Безкоровайний В.В.		14.12.21		
Н. контроль	Безкоровайний В.В.		14.12.21		
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			Кафедра системотехніки	Аркушів 1

Схема генетичного алгоритму



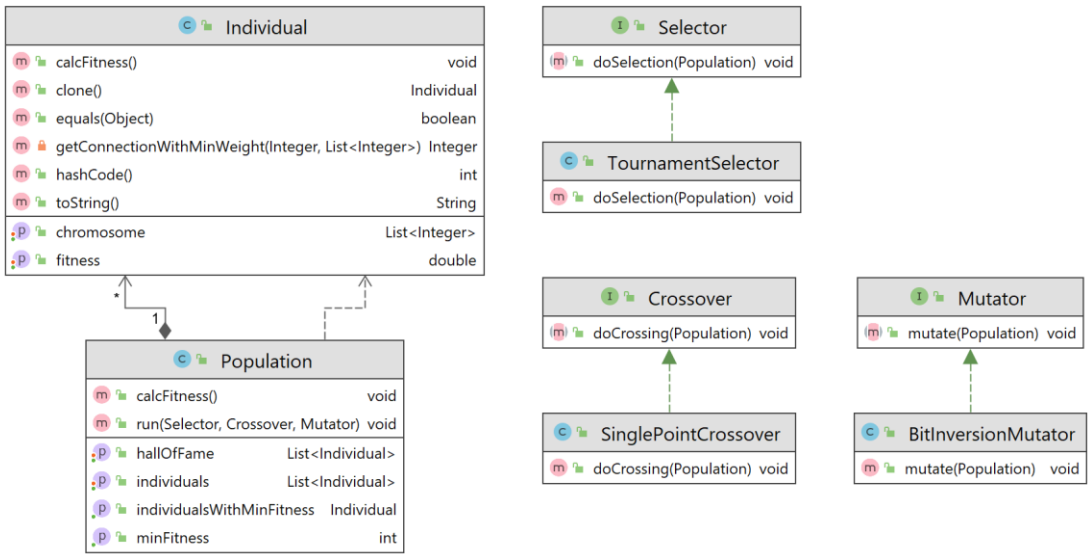
Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*

Діаграма варіантів використання системи



Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*

Діаграма класів системи



Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*

Екранні форми вводу даних

Chromosome Length

Scale

Cost Per 1 Km Of Canal

Cost Per 1 Node

Main Node Cost

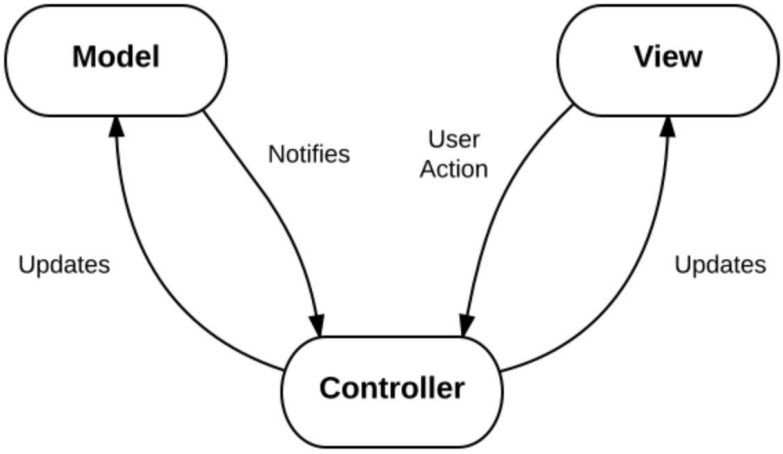
Additional Node Cost

Submit

№	X, mm	Y, mm
1		
2		
3		
4		
5		

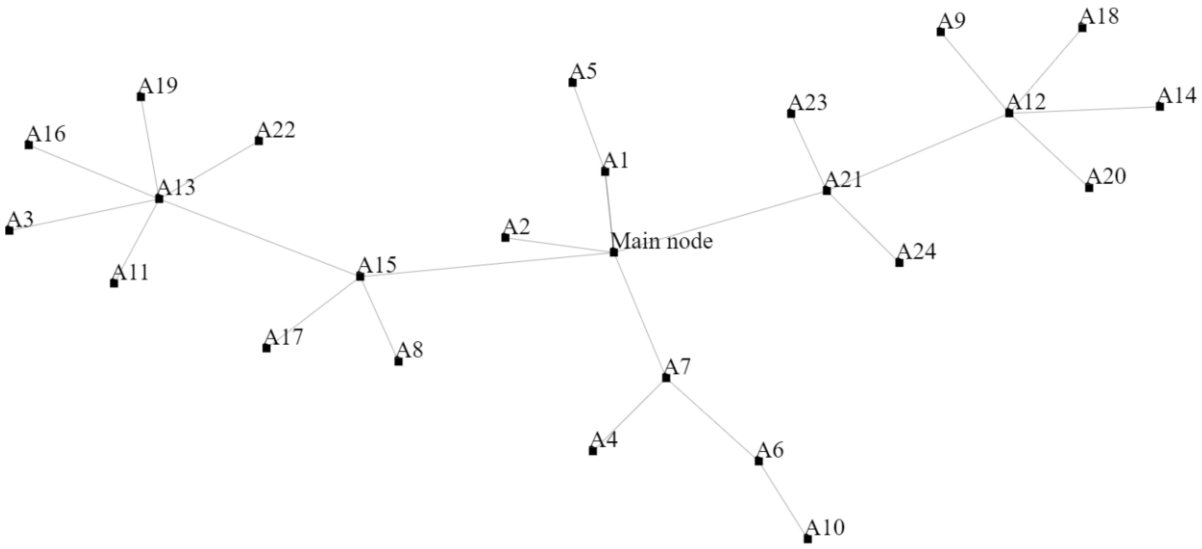
Розробив	Ляскovка Я.І.		14.12.21	Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж	
Перевірив	Безкоровайний В.В.		14.12.21		
Н. контроль	Безкоровайний В.В.		14.12.21		
				СПРм-20-1	Аркуш 1
Затвердив	Гребеннік І.В.			Кафедра системотехніки	Аркушів 1

Схема роботи додатку за архітектурою MVC

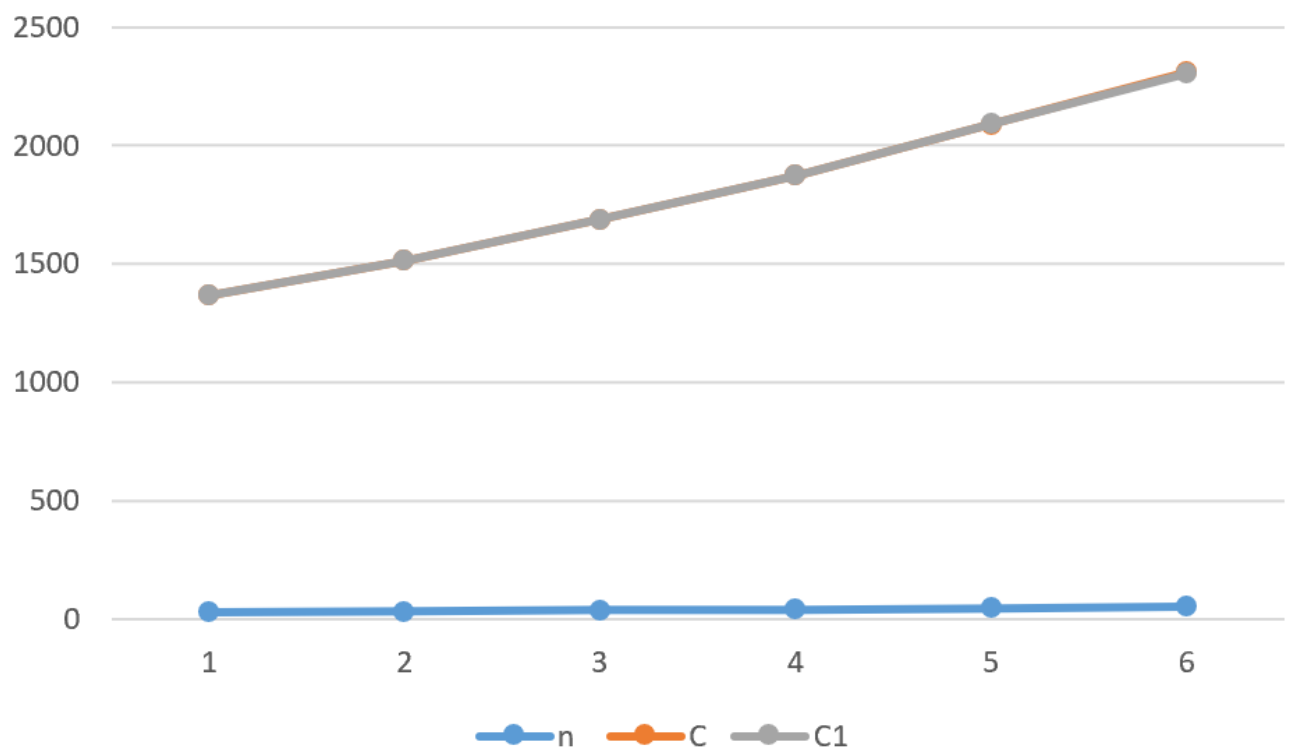


Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*

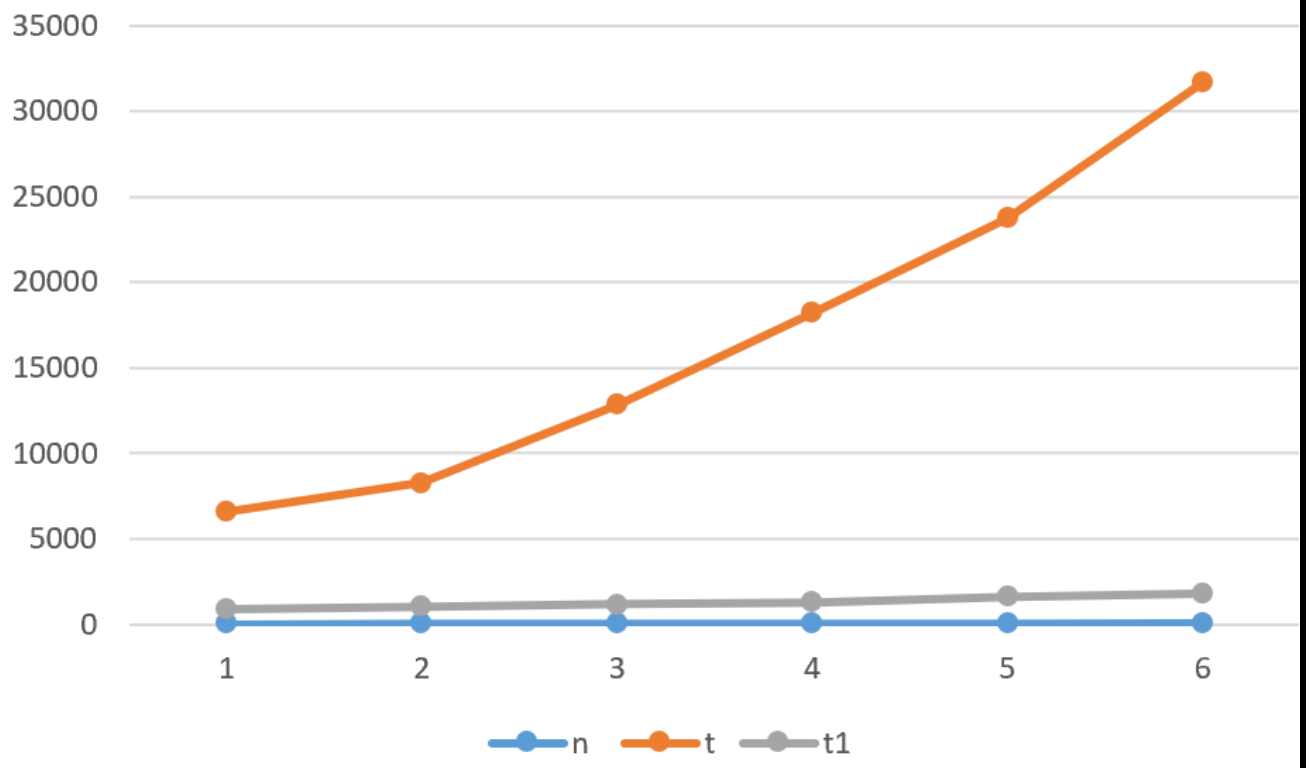
Екранна форма виводу результатів



Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*



Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*



Розробив	*Лясковка Я.І.*		*14.12.21*	*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*	
Перевірив	*Безкоровайний В.В.*		*14.12.21*		
Н. контроль	*Безкоровайний В.В.*		*14.12.21*		
				СПРм-20-1	*Аркуш 1*
Затвердив	*Гребеннік І.В.*			*Кафедра системотехніки*	*Аркушів 1*

Додаток Б

Текст програми

ГЮИК. 401210.007-01 12 01

(позначення документа)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

«ЗАТВЕРДЖУЮ»
Керівник кваліфікаційної роботи
_____ проф. Безкоровайний В.В.
(підпис)

ДОСЛІДЖЕННЯ МЕТОДІВ ПІДТРИМКИ ПРИЙНЯТТЯ ПРОЕКТНИХ РІШЕНЬ З РЕІНЖИНІРИНГУ КОРПОРАТИВНИХ КОМП'ЮТЕРНИХ МЕРЕЖ

Текст програми

ЛИСТ ЗАТВЕРЖДЕНИЯ

ГЮИК.401210.007-01 12 01-АЗ

РОЗРОБИВ;
Ст. гр. СПРм-20-1
Лясковка Я.І.

2021

ЗАТВЕРДЖЕНО
ГЮИК.401210.007-01 12 01 АЗ

ДОСЛІДЖЕННЯ МЕТОДІВ ПІДТРИМКИ ПРИЙНЯТТЯ ПРОЕКТНИХ РІШЕНЬ З РЕІНЖИНІРИНГУ КОРПОРАТИВНИХ КОМП'ЮТЕРНИХ МЕРЕЖ

Текст програми

ГЮИК.401210.007-01 12 01 АЗ

Аркушів 9

2021

```
 package ua.nure.gaserver.entities;

import java.util.*;

import static ua.nure.gaserver.configurations.Configuration.*;

public class Individual implements Cloneable {
    private List<Integer> chromosome;
    private double fitness;
    private List<Integer> nodePositions;

    public Individual() {
        chromosome = new ArrayList<>(CHROMOSOME_LENGTH);
        for (int i = 0; i < CHROMOSOME_LENGTH; i++) {
            int val;
            if (i == MAIN_NODE_INDEX) {
                val = 1;
            } else {
                val = RANDOM.nextInt(2);
            }
            chromosome.add(val);
        }
    }

    @Override
    public Individual clone() {
        Individual individual = new Individual();
        individual.setChromosome(new ArrayList<>(chromosome));
        individual.setNodePositions(new ArrayList<>(nodePositions));
        individual.setFitness(fitness);
        return individual;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Individual that = (Individual) o;
        return chromosome.equals(that.chromosome);
    }

    @Override
    public int hashCode() {
        return Objects.hash(chromosome);
    }

    @Override
    public String toString() {
        return chromosome.toString();
```

```
    public void calcFitness() {
        double fitness = MAIN_NODE_COST;
        nodePositions = new ArrayList<>();
        for (int i = 0; i < chromosome.size(); i++) {
            if (chromosome.get(i) == 1) nodePositions.add(i);
        }
        int numberOfAdditionalNodes = nodePositions.size() - 1;
        fitness += numberOfAdditionalNodes * COST_PER_1_NODE;
        fitness += (CHROMOSOME_LENGTH - 1) * ADDITIONAL_NODE_COST;

        Map<Integer, Integer> connections = getConnections(nodePositions);

        for (Map.Entry<Integer, Integer> entry : connections.entrySet()) {
            Integer i = entry.getKey();
            Integer j = entry.getValue();
            fitness += ADJACENCY_MATRIX[i][j] * COST_PER_1_KM_OF_CANAL;
        }
        this.fitness = fitness;
    }

    public static Map<Integer, Integer> getConnections(List<Integer> nodePositions) {
        Map<Integer, Integer> connections = new HashMap<>();
        setNodeEdgesWithMinWeight(nodePositions, connections);
        for (int i = 0; i < CHROMOSOME_LENGTH; i++) {
            if (connections.containsKey(i)) continue;
            setAdditionalNodeEdgesWithMinWeight(i, nodePositions, connections);
        }
        return connections;
    }

    public static void setNodeEdgesWithMinWeight(List<Integer> nodePositions, Map<Integer,
Integer> connections) {
        for (int i = 0; i < nodePositions.size(); i++) {
            Integer source = nodePositions.get(i);
            int minIndex = 0;
            double minWeight = Double.MAX_VALUE;
            for (int j = i+1; j < nodePositions.size(); j++) {
                Integer destination = nodePositions.get(j);
                if (connections.containsValue(source) || connections.containsValue(destination))
continue;
                double weight = ADJACENCY_MATRIX[source][destination];
                if (weight < minWeight) {
                    minIndex = destination;
                    minWeight = weight;
                }
            }
            connections.put(source, minIndex);
        }
    }
```

```
    public static void setAdditionalNodeEdgesWithMinWeight(Integer i, List<Integer> nodePositions,
                                                            Map<Integer, Integer> connections) {
        int minIndex = 0;
        double minWeight = Double.MAX_VALUE;
        for (int j = 0; j < nodePositions.size(); j++) {
            Integer destination = nodePositions.get(j);
            if (i.equals(destination)) continue;
            double weight = ADJACENCY_MATRIX[i][destination];
            if (weight < minWeight) {
                minIndex = destination;
                minWeight = weight;
            }
        }
        connections.put(i, minIndex);
    }

    public double getFitness() {
        return fitness;
    }

    public void setFitness(double fitness) {
        this.fitness = fitness;
    }

    public List<Integer> getChromosome() {
        return chromosome;
    }

    public void setChromosome(List<Integer> chromosome) {
        this.chromosome = chromosome;
    }

    public List<Integer> getNodePositions() {
        return nodePositions;
    }

    public void setNodePositions(List<Integer> nodePositions) {
        this.nodePositions = nodePositions;
    }
}
package ua.nure.gaserver.entities;

import ua.nure.gaserver.crossing.Crossover;
import ua.nure.gaserver.json.Statistic;
import ua.nure.gaserver.mutators.Mutator;
import ua.nure.gaserver.selections.Selector;

import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.Stream;
```

```
import static ua.nure.gaserver.configurations.Configuration.*;
public class Population {
    private List<Individual> individuals;
    private List<Individual> hallOfFame;

    public Population() {
        this.individuals = Stream.generate(Individual::new)
                                 .limit(POPULATION_SIZE)
                                 .collect(Collectors.toList());
    }

    public Statistic run(Selector selector, Crossover crossover, Mutator mutator) {
        long l = System.currentTimeMillis();
        double previousMinFitness;
        int duplicateFitnessCounter = 0;
        int generationCounter = 0;
        doRestriction();
        calcFitness();
        System.out.println("Generation " + generationCounter);
        previousMinFitness = getMinFitness();
        System.out.println("Min fitness " + previousMinFitness);
        while (generationCounter < MAX_GENERATIONS /*&& duplicateFitnessCounter <
MAX_DUPLICATE_FITNESS_COUNTER*/) {
            generationCounter++;
            selector.doSelection(this);
            crossover.doCrossing(this);
            mutator.mutate(this);
            doRestriction();
            calcFitness();
            System.out.println("Generation " + generationCounter);
            double minFitness = getMinFitness();
            if (previousMinFitness == minFitness) {
                duplicateFitnessCounter++;
            } else {
                previousMinFitness = minFitness;
                duplicateFitnessCounter = 0;
            }
            System.out.println("Min fitness " + previousMinFitness);
        }
        return new Statistic(previousMinFitness, System.currentTimeMillis() - l,
getIndividualWithMinFitness());
    }

    private void doRestriction() {
        if (MAX_NUMBER_OF_NODES != 0) {
            individuals.forEach(individual -> {
                List<Integer> nodePositions = new ArrayList<>();
                for (int i = 0; i < CHROMOSOME_LENGTH; i++) {
                    if (individual.getChromosome().get(i) == 1) nodePositions.add(i);
                }
```

```
            int size = nodePositions.size();
            if (size > MAX_NUMBER_OF_NODES) {
                Set<Integer> pointsSet = new HashSet<>();
                while (pointsSet.size() < size - MAX_NUMBER_OF_NODES) {
                    int point = RANDOM.ints(1, 1, size).findFirst().orElse(0);
                    pointsSet.add(point);
                }
                pointsSet.forEach(value -> {
                    individual.getChromosome().set(nodePositions.get(value), 0);
                });
                individual.calcFitness();
            }
        });
    }
}

public void calcFitness() {
    individuals.forEach(Individual::calcFitness);
}

public double getMinFitness() {
    return individuals
            .stream()
            .map(Individual::getFitness)
            .min(Double::compareTo)
            .orElse(0d);
}

public Individual getIndividualWithMinFitness() {
    return individuals
            .stream()
            .min(Comparator.comparingDouble(Individual::getFitness))
            .orElse(null);
}

public List<Individual> getIndividuals() {
    return individuals;
}

public List<Individual> getHallOfFame() {
    return hallOfFame;
}

public void setHallOfFame(List<Individual> hallOfFame) {
    this.hallOfFame = hallOfFame;
}

public void setIndividuals(List<Individual> individuals) {
    this.individuals = individuals;
}
```

```
}


package ua.nure.gaserver.selections;

import ua.nure.gaserver.entities.Individual;
import ua.nure.gaserver.entities.Population;

import java.util.*;
import java.util.stream.Collectors;

import static ua.nure.gaserver.configurations.Configuration.*;

public class TournamentSelector implements Selector {
    @Override
    public void doSelection(Population population) {
        List<Individual> individuals = population.getIndividuals();
        List<Individual> offspring = new ArrayList<>(POPULATION_SIZE);
        int i = 0;
        if (ENABLE_ELITISM) {
            i += HALL_OF_FAME_SIZE;
            List<Individual> hallOfFame = individuals.stream()
                    .sorted(Comparator.comparingDouble(Individual::getFitness))
                    .limit(HALL_OF_FAME_SIZE)
                    .peek(Individual::clone)
                    .collect(Collectors.toList());
            offspring.addAll(hallOfFame);
            population.setHallOfFame(hallOfFame);
        }
        for (; i < POPULATION_SIZE; i++) {
            Set<Integer> indexes = new HashSet<>();
            while (indexes.size() < 3) {
                int point = RANDOM.nextInt(POPULATION_SIZE);
                indexes.add(point);
            }

            Individual individual = indexes
                    .stream()
                    .map(individuals::get)
                    .min(Comparator.comparingDouble(Individual::getFitness))
                    .orElse(null);
            offspring.add(individual.clone());
        }
        population.setIndividuals(offspring);
    }
}
package ua.nure.gaserver.crossing;

import ua.nure.gaserver.entities.Individual;
import ua.nure.gaserver.entities.Population;
```

```
import java.util.List;

import static ua.nure.gaserver.configurations.Configuration.*;
public class SinglePointCrossover implements Crossover {
    @Override
    public void doCrossing(Population population) {
        for (int i = 0; i < POPULATION_SIZE; i += 2) {
            Individual individual = population.getIndividuals().get(i);
            Individual individual1 = population.getIndividuals().get(i + 1);
            if (ENABLE_ELITISM && (population.getHallOfFame().contains(individual) ||
population.getHallOfFame().contains(individual1)))
                continue;
            if (RANDOM.nextDouble() < P_CROSSOVER) {
                int point = RANDOM.ints(1, 1, CHROMOSOME_LENGTH)
                        .findFirst()
                        .orElse(0);
                List<Integer> tail = individual1.getChromosome().subList(point,
CHROMOSOME_LENGTH);
                for (int j = point; j < CHROMOSOME_LENGTH; j++) {
                    individual1.getChromosome().set(j, individual.getChromosome().get(j));
                    individual.getChromosome().set(j, tail.get(j - point));
                }
            }
        }
    }
}
package ua.nure.gaserver.mutators;

import ua.nure.gaserver.entities.Population;

import java.util.List;

import static ua.nure.gaserver.configurations.Configuration.*;

public class BitInversionMutator implements Mutator {
    @Override
    public void mutate(Population population) {
        population.getIndividuals().forEach(individual -> {
            if (ENABLE_ELITISM && population.getHallOfFame().contains(individual)) return;
            if (RANDOM.nextDouble() < P_MUTATION) {
                List<Integer> chromosome = individual.getChromosome();
                for (int i = 0; i < chromosome.size(); i++) {
                    if (MAIN_NODE_INDEX == i) continue;
                    if (RANDOM.nextDouble() < P_MUTATION_OF_GEN) {
                        chromosome.set(i, chromosome.get(i) == 0 ? 1 : 0);
                    }
                }
            }
        });
    }
```

```
}
package ua.nure.gaserver.services;

import org.springframework.stereotype.Service;
import ua.nure.gaserver.crossing.Crossover;
import ua.nure.gaserver.crossing.SinglePointCrossover;
import ua.nure.gaserver.entities.Population;
import ua.nure.gaserver.json.GraphRequest;
import ua.nure.gaserver.json.Statistic;
import ua.nure.gaserver.mutators.BitInversionMutator;
import ua.nure.gaserver.mutators.Mutator;
import ua.nure.gaserver.selections.Selector;
import ua.nure.gaserver.selections.TournamentSelector;
import ua.nure.gaserver.utils.DrawGraph;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Stream;

import static ua.nure.gaserver.configurations.Configuration.*;

@Service
public class GraphService {
    public Map<String, String> getGraph(GraphRequest graphRequest) {
        long l = System.currentTimeMillis();
        COORDINATES = new double[50][2];
        for (int i = 0; i < 50; i++) {
            COORDINATES[i][0] = 80.0 + (360.0 - 80.0) * RANDOM.nextDouble();
            COORDINATES[i][1] = 80.0 + (360.0 - 80.0) * RANDOM.nextDouble();
        }
        initAdjacencyMatrix();
        List<Statistic> statistics = new ArrayList<>();
        Population population = new Population();
//        MAX_NUMBER_OF_NODES = 0;
//        population.run(new TournamentSelector(), new SinglePointCrossover(), new
BitInversionMutator());
//        for (int i = 0; i < CHROMOSOME_LENGTH; i++) {
//            MAX_NUMBER_OF_NODES +=i+1;
//            population = new Population();
//            Statistic statistic = population.run(new TournamentSelector(), new
SinglePointCrossover(), new BitInversionMutator());
//            statistics.add(statistic);
//            System.out.println("Solution found with min fitness " + statistic.getMinFitness());
//            System.out.println("Method execution took " + statistic.getExecutionTime() + " ms
with max nodes " + MAX_NUMBER_OF_NODES);
//        }

        for (int i = 0; i < 6; i++) {
            CHROMOSOME_LENGTH = 25;
```

```
            CHROMOSOME_LENGTH += i*5;
                            population = new Population();
                Statistic statistic = population.run(new TournamentSelector(), new
SinglePointCrossover(), new BitInversionMutator());
                statistics.add(statistic);
//            statistics.clear();
//            for (int j = 0; j < CHROMOSOME_LENGTH; j++) {
//                MAX_NUMBER_OF_NODES +=i+1;
//                population = new Population();
//                Statistic statistic = population.run(new TournamentSelector(), new
SinglePointCrossover(), new BitInversionMutator());
//                statistics.add(statistic);
//                System.out.println("Solution found with min fitness " +
statistic.getMinFitness());
//                System.out.println("Method execution took " + statistic.getExecutionTime() + "
ms with max nodes " + MAX_NUMBER_OF_NODES);
//            }
        }

        System.out.println("Method execution took " + (System.currentTimeMillis() - l) + " ms");
        return DrawGraph.draw(population.getIndividualWithMinFitness());
    }

    private static void initAdjacencyMatrix() {
        for (int i = 0; i < COORDINATES.length; i++) {
            for (int j = i; j < COORDINATES.length; j++) {
                double[] a = COORDINATES[i];
                double[] b = COORDINATES[j];
                double distance = Math.sqrt(Math.pow(b[0] - a[0], 2) + Math.pow(b[1] - a[1], 2));
                distance *= SCALE;
                distance *= COEFFICIENT_TO_CONVERT_DISTANCE_IN_KM;
                ADJACENCY_MATRIX[i][j] = distance;
                ADJACENCY_MATRIX[j][i] = distance;
            }
        }
    }
}
```

Додаток В

Заява щодо самостійності виконання роботи та можливості її публікації

ЗАЯВА
щодо самостійності виконання письмової роботи

Я, *Лясковка Ян Ігорович*
(прізвище, ім'я, по батькові)

посада *студент групи СПРм-20-1*
кафедра *Системотехніки*

заявляю: моя письмова робота на тему «*Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж*»
(назва роботи)

представлена в *екзаменаційну комісію.*
(спеціалізовану вчену раду, екзаменаційну комісію тощо)

Для публічного захисту, виконана самостійно і в ній не міститься елементів плагіату. Всі запозичені з друкованих та електронних джерел, а також із раніше виконаних дослідницьких робіт та захищених кандидатських і докторських дисертацій мають відповідні посилання.

Я Ознайомлений (а) з діючим положенням «Про протидію плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску письмової роботи до захисту та застосування дисциплінарних заходів.

09.06.2020 р.
(Дата)

(Підпис)

Додаток Г

Протокол перевірки тексту пояснювальної записки електронною системою на плагіат

UNICHECK

Имя пользователя:
Кісель Тетяна Іванівна каф. СТ

ID проверки:
1009709904

Дата проверки:
17.12.2021 22:32:27 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
17.12.2021 22:33:30 EET

ID пользователя:
94940

Название файла: 2021_М_СТ_СПРм-20-1_Лясковка_Я_І

Количество страниц: 66 Количество слов: 12466 Количество символов: 91668 Размер файла: 1.68 MB ID файла: 1009708358

9.59% Совпадения

Наибольшее совпадение: 5.02% с Интернет-источником (http://www.economy.nayka.com.ua/?op=1&z=943)

6.81% Источники из Интернета 52 ..... Страница 68

3.41% Источники из Библиотеки 67 ..... Страница 68

0.4% Цитат

Цитаты 1 ..... Страница 69

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы 46

Додаток Д

Відомість кваліфікаційної роботи

ГЮИК.401210.007 ДЗ

(позначення документа)

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1.	ГЮИК.401210.007 ПЗ	Пояснювальна записка	77 с.
2.	ГЮИК.401210.007-01 12 01	Текст програми	9 с.
		Графічні документи	
3.		Схема зв'язку категорій "елементи",	
		"відношення", "топологія" та	
		"властивості"	1 аркуш
4.		Схема генетичного алгоритму	1 аркуш
5.		Діаграма варіантів використання	
		системи	1 аркуш
6.		Діаграма класів системи	1 аркуш
7.		Екранні форми вводу даних	1 аркуш
8.		Схема роботи додатку за	
		архітектурою MVC	1 аркуш
9.		Екранна форма виводу результатів	1 аркуш
10.		Залежність витрат від кількості	
		вузлів в мережі	1 аркуш
11.		Залежність часу розв'язання задачі	
		від кількості вузлів мережі	1 аркуш

Змін.	Аркуш	№ документа	Підпис	Дата	ГЮИК.401210.007 ДЗ			
Розробив		Лясковка Я.І.		14.12.21	Дослідження методів підтримки прийняття проектних рішень з реінжинірингу корпоративних комп'ютерних мереж	Літера	Аркуш	Аркушів
Перевірив		Безкоровайний В.В.		14.12.21		у	1	1
Н. контроль		Безкоровайний В.В.		14.12.21		ХНУРЕ Кафедра СТ		
Затвердив		Гребеннік І.В.			Відомість атестаційної роботи			