|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНС-13 | 5 | ГА ДЛЯ ЗАДАЧІ КОМІВОЯЖЕРА (TSP) |  |  |
| Назар Б.В. | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

1. **Мета роботи**

Ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації.

1. **Індивідуальне завдання**

**Проблема полягає в наступному.** У вас є безліч міст (представлені у вигляді точок на площині з X і Y координати). Мета полягає в тому, щоб знайти найкоротший маршрут, який відвідує кожне місто рівно один раз, повертаючись в кінці своєї відправної точки. Дано від 10 до 50 точок. Мова програмування довільна.

Розробити програму для вирішення задачі комівояжера (TSP).

**Варіант № 13 (1)** Турнірна селекція.

1. **Програмна реалізація завдання**

**Фрагменти коду на мові програмування Java:**

**TSP.class:**

**package** tsp;

**import** java.awt.Graphics2D;

**import** java.util.Date;

**import** javax.swing.SwingUtilities;

**public** **class** TSP {

**static** **int** *nsity*=45; //Кількість міст

**static** **int** *npop*=350; // Розмір популяції

**static** **int** [] *x*=**new** **int**[*nsity*];

**static** **int** [] *y*=**new** **int**[*nsity*];

**static** **int** [] *xline*= **new** **int**[*nsity*];

**static** **int** [] *yline*= **new** **int**[*nsity*];

**public** **static** **void** main(String[] args) {

System.***out***.println("Кординати міст:");

**for** (**int** i = 0; i < *nsity* ; i++)

{

**int** newx = (**int**)(Math.*random*()\*200);

*x*[i]=newx;

**int** newy = (**int**)(Math.*random*()\*200);

*y*[i]=newy;

System.***out***.print("["+*x*[i]+","+*y*[i]+"],");

**if**(i%10==9 ){

System.***out***.println();

}

}

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

TourManager.*addCity*(**new** City(*x*[i],*y*[i]));

}

Population pop = **new** Population(*npop*, **true**);

System.***out***.println("\nРозмір популяції: "+*npop*);

Date currentTimeBefore = **new** Date();

**long** timeBefore = currentTimeBefore.getTime();

pop = GA.*evolvePopulation*(pop);

**for** (**int** i = 0; i < 100; i++) {

pop = GA.*evolvePopulation*(pop);

}

Date currentTimeAfter = **new** Date();

**long** timeAfter= currentTimeAfter.getTime();;

**long** time = timeAfter-timeBefore;

System.***out***.println("Довжина шляху: " + pop.getFittest().getDistance());

System.***out***.println("Час виконання: " + time + " мс");

System.***out***.println("Знайдений шлях:");

Tour t=**new** Tour();

t=pop.getFittest();

**for**(**int** i=0;i<*nsity*;i++){

*xline*[i]= t.getCity(i).x;

*yline*[i]= t.getCity(i).y;

}

**for** (**int** i = 0; i < *nsity* ; i++)

{

System.***out***.print("["+*xline*[i]+","+*yline*[i]+"],");

**if**(i%10==9 ){

System.***out***.println();

}

}

SwingUtilities.*invokeLater*(**new** Runnable() {

@Override

**public** **void** run() {

CartesianFrame frame = **new** CartesianFrame();

frame.showUI();;

}

});

}

}

**Class Ga:**

**public** **class** GA {

**private** **static** **final** **double** ***mutationRate*** = 0.015;

**private** **static** **final** **int** ***tournamentSize*** = 5;

**private** **static** **final** **boolean** ***elitism*** = **true**;

**public** **static** Population evolvePopulation(Population pop) {

Population newPopulation = **new** Population(pop.populationSize(), **false**);

**int** elitismOffset = 0;

**if** (***elitism***) {

newPopulation.saveTour(0, pop.getFittest());

elitismOffset = 1;

}

**for** (**int** i = elitismOffset; i < newPopulation.populationSize(); i++) {

Tour parent1 = *tournamentSelection*(pop);

Tour parent2 = *tournamentSelection*(pop);

Tour child = *crossover*(parent1, parent2);

newPopulation.saveTour(i, child);

}

**for** (**int** i = elitismOffset; i < newPopulation.populationSize(); i++) {

*mutate*(newPopulation.getTour(i));

}

**return** newPopulation;

}

**public** **static** Tour crossover(Tour parent1, Tour parent2) {

Tour child = **new** Tour();

**int** p1 = (**int**) (Math.*random*() \* parent1.tourSize());

**int** p2 = (**int**) (Math.*random*() \* parent1.tourSize());

**for**(**int** i = 0; i < child.tourSize(); i++)

{

**if**(i>=p1 && i<=p2) child.setCity(i, **null**);

**else** child.setCity(i, parent1.getCity(i));

}

**int** n = 0;

**for** ( **int** j = 0; j < parent2.tourSize(); j++)

{

**boolean** t = **false**;

**for** ( **int** k = 0; k < child.tourSize(); k++)

{

**if**(parent2.getCity(j) == child.getCity(k)) {

t = **true**;

**break**;

}

}

**if** (t== **false**){

child.setCity(p1+n, parent2.getCity(j));

n = n+1;

}

}

**return** child;

}

**private** **static** **void** mutate(Tour tour) {

**int** tourPos1 = (**int**) (tour.tourSize() \* Math.*random*());

**int** tourPos2 = (**int**) (tour.tourSize() \* Math.*random*());

**if** (tourPos2 < tourPos1) {

**int** q = tourPos1;

tourPos1 = tourPos2;

tourPos2 = q;

}

**for**(**int** i = 0; i <= (tourPos2 - tourPos1 +1)/2 ; i++)

{

City q = tour.getCity(tourPos1+i);

tour.setCity(tourPos1+i, tour.getCity(tourPos2-i));

tour.setCity(tourPos2-i, q);

}

}

**private** **static** Tour tournamentSelection(Population pop) {

Population truncation = **new** Population(***tournamentSize***, **false**);

**for** (**int** i = 0; i < ***tournamentSize***; i++) {

**int** randomId = (**int**) (Math.*random*() \* pop.populationSize());

truncation.saveTour(i, pop.getTour(randomId));

}

Tour fittest = truncation.getFittest();

**return** fittest;

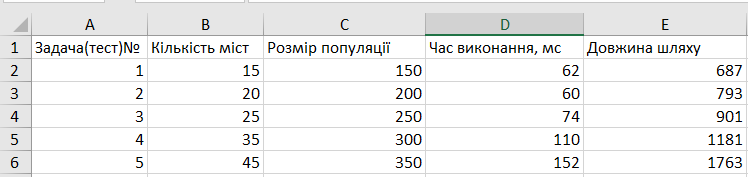
}

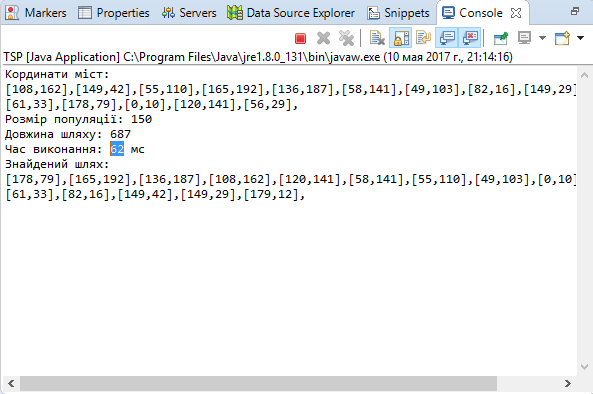
}

**Результати роботи програми:**

*Таблиця 1*

*Результатами роботи програми з різними параметрами*





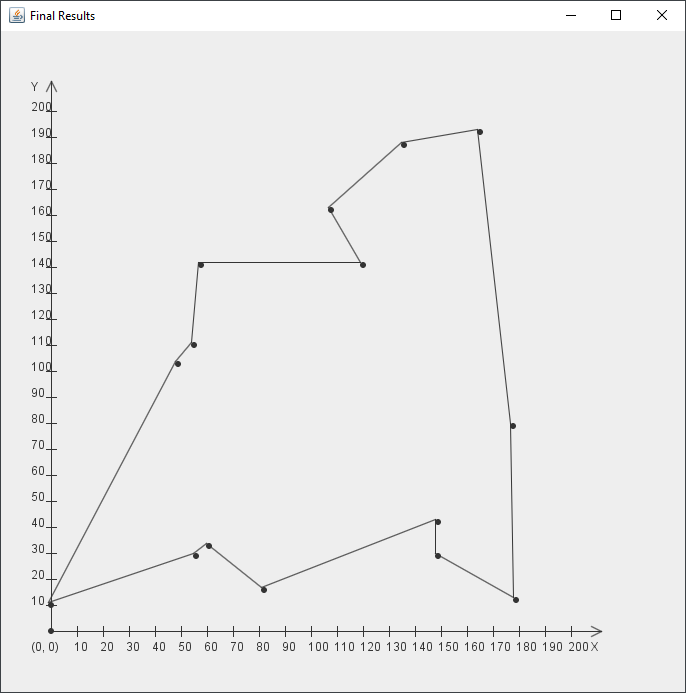
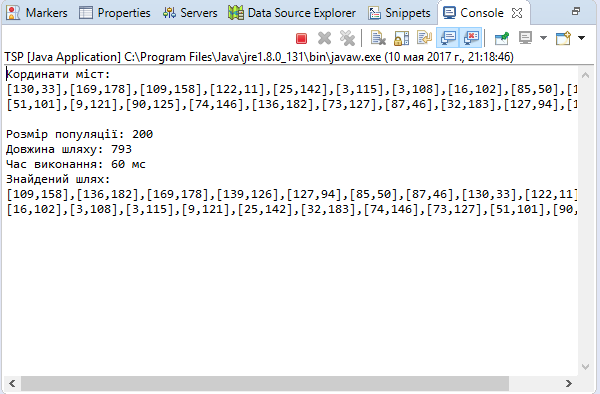


Рис. 1 Результати при запуску програми з даними для Тесту №1



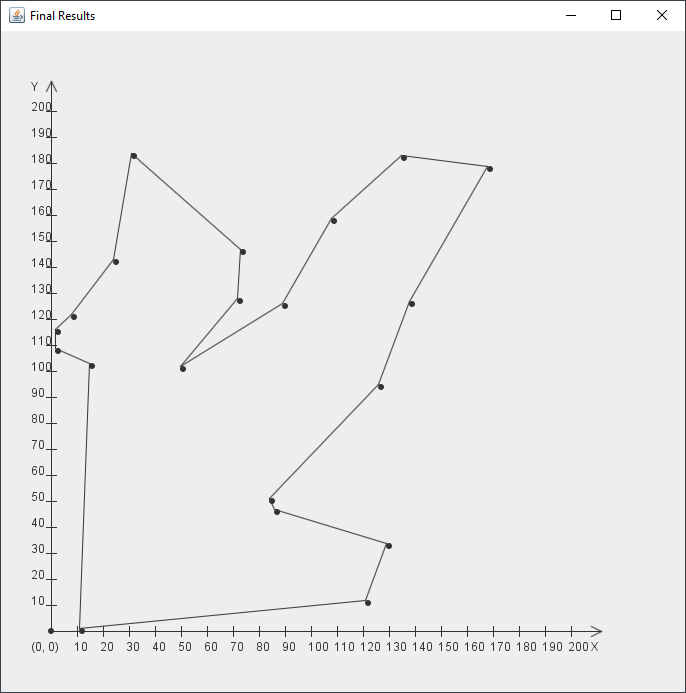
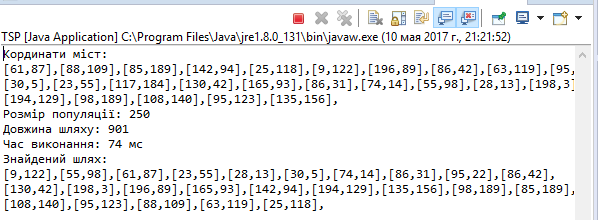


Рис. 2 Результати при запуску програми з даними для Тесту №2



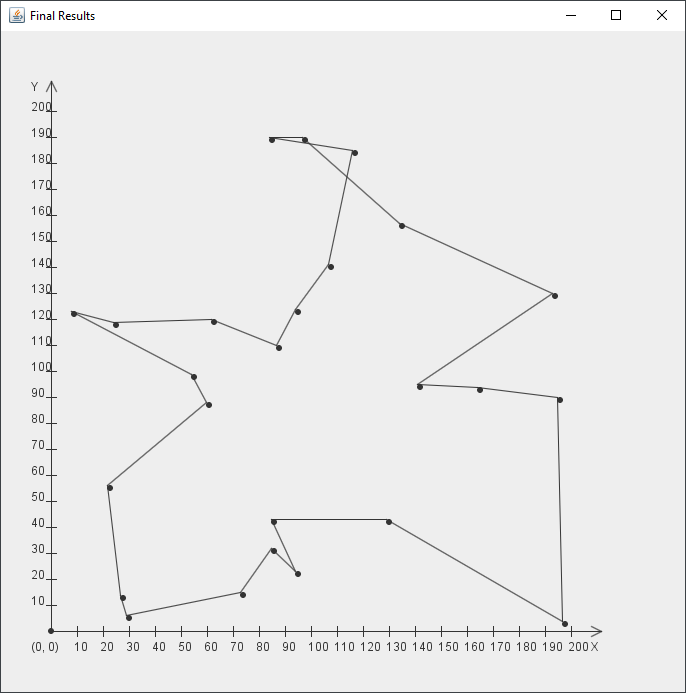
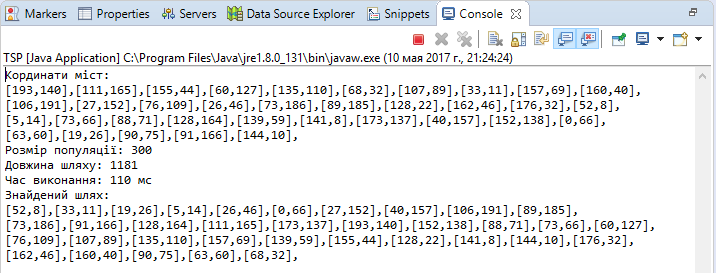


Рис. 3 Результати при запуску програми з даними для Тесту №3



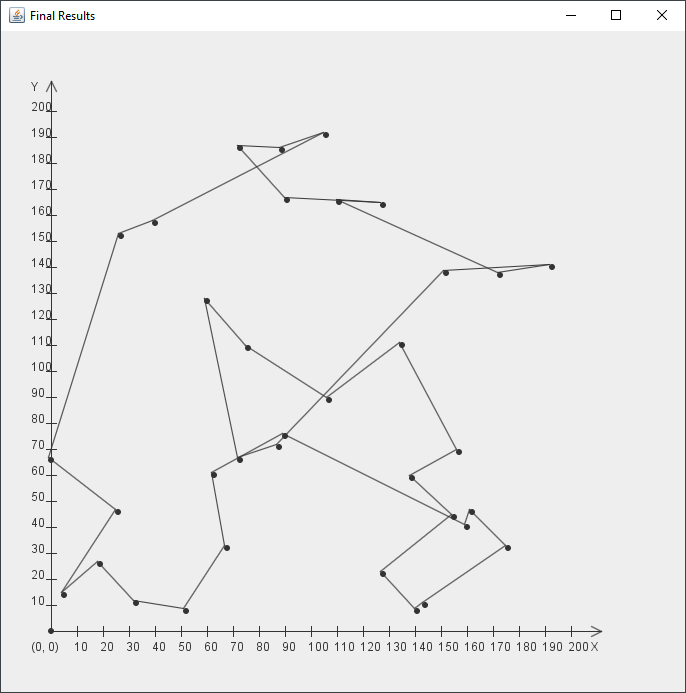
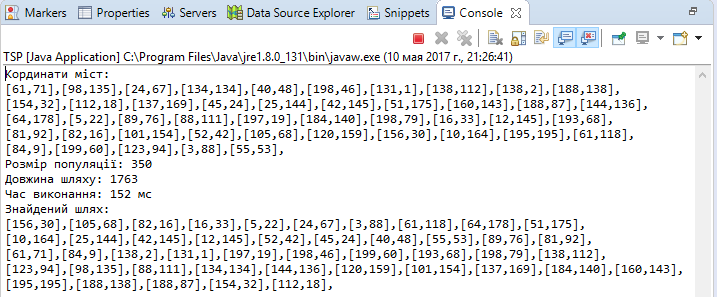


Рис. 4 Результати при запуску програми з даними для Тесту №4



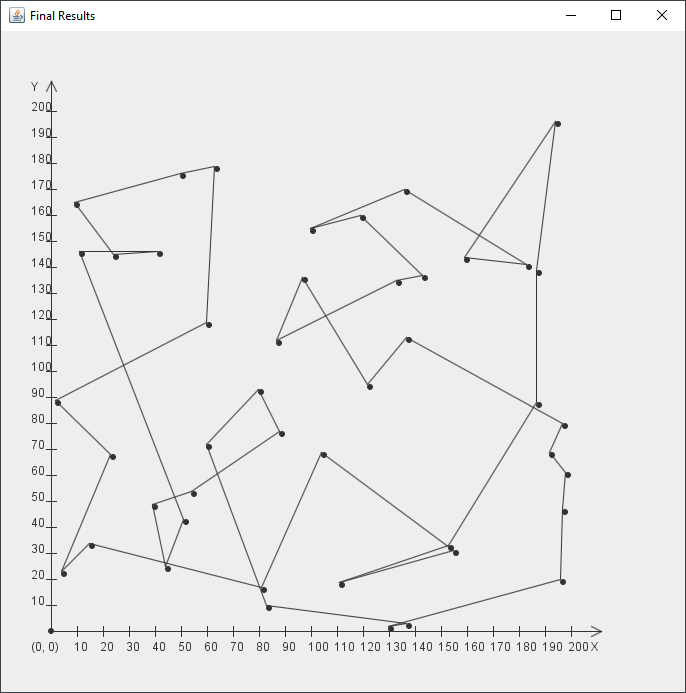


Рис. 5 Результати при запуску програми з даними для Тесту №5

1. **Висновки**

У результаті виконання лабораторної роботи я розробив програму, яка реалізовує генетичний алгоритм для розв’язання задачі комівояжера.