*дата :*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНC-13 | 5 (номер лаб.) | Запрограмувати ГА для задачі комівояжера |  |  |
| Абрам Б.Я. | |
| № залікової: 1608401 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р. З. | |

**Мета:** Розробити за допомогою Microsoft Visual Studio 2013 програмне забезпечення для вирішення задачі комівояжера.

**Теоретичні відомості**

### 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.

### Турнірний відбір(селекція). При турнірному відборі (tournament selection) з популяції, яка складається із {\displaystyle N} особин, вибираються випадковим чином {\displaystyle t} особин, і найкраща особина записується в проміжний масив. Ця операція повторюється {\displaystyle N} раз. Особини в отриманому проміжному масиві потім використовуються для схрещування (також випадковим чином). Розмір групи рядків, що відбираються для турніру, часто дорівнює 2. У цьому випадку говорять про двійковий (парний) турнір. Взагалі ж {\displaystyle t} називають чисельністю турніру. Перевагою даного способу є те, що він не вимагає додаткових обчислень.

**Індивідуальне завдання**

Викорстати метод турнірної селекції.

**Фрагменти коду програми**

using System;

using AForge.Genetic;

namespace TSP{

public class TSPChromosome : PermutationChromosome{

private int[,] map = null;

public TSPChromosome( int[,] map ) : base( map.GetLength( 0 ) ){

this.map = map;}

protected TSPChromosome( TSPChromosome source ) : base( source ){

this.map = source.map;}

//Створення нової(випадкової) хромосоми

public override IChromosome CreateOffspring( ){

return new TSPChromosome( map );}

//Створення копії хромосоми

public override IChromosome Clone( ){

return new TSPChromosome( this );}

//Оператор схрещування

public override void Crossover( IChromosome pair ){

TSPChromosome p = (TSPChromosome) pair;

// перевірка правильності пари

if ( ( p != null ) && ( p.length == length ) ){

ushort[] child1 = new ushort[length];

ushort[] child2 = new ushort[length];

// створення двох дітей

CreateChildUsingCrossover( this.val, p.val, child1 );

CreateChildUsingCrossover( p.val, this.val, child2 );

// заміна батьків дітьми

this.val = child1;

p.val = child2;} }

// створення нової дитини використовуючи кросовер її батьків

private void CreateChildUsingCrossover( ushort[] parent1, ushort[] parent2, ushort[] child ){

// тимчасовий масив, щоб вказати, якщо певний ген вже присутній в дитині

bool[] geneIsBusy = new bool[length];

// попередній ген у дитини і двох наступних кандидатів

ushort prev, next1, next2;

// доступність кандидата - кандидат є доступним,якщо він ще не дитина

bool valid1, valid2;

int j, k = length - 1;

// перший ген дитини береться від другого з батьків

prev = child[0] = parent2[0];

geneIsBusy[prev] = true;

// заповнення всіх інших генів дитини

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

// знайти наступний ген після попереднього в обох батьків

// 1

for ( j = 0; j < k; j++ ){

if ( parent1[j] == prev )

break;}

next1 = ( j == k ) ? parent1[0] : parent1[j + 1];

// 2

for ( j = 0; j < k; j++ ){

if ( parent2[j] == prev )

break;}

next2 = ( j == k ) ? parent2[0] : parent2[j + 1];

// перевірка генів кандидатів на доступність

valid1 = !geneIsBusy[next1];

valid2 = !geneIsBusy[next2];

// вибір гену

if ( valid1 && valid2 ){

// обидва кандидати доступні

// вибір одного найближчого міста

double dx1 = map[next1, 0] - map[prev, 0];

double dy1 = map[next1, 1] - map[prev, 1];

double dx2 = map[next2, 0] - map[prev, 0];

double dy2 = map[next2, 1] - map[prev, 1];

prev = ( Math.Sqrt( dx1 \* dx1 + dy1 \* dy1 ) < Math.Sqrt( dx2 \* dx2 + dy2 \* dy2 ) ) ? next1 : next2; }

else if ( !( valid1 || valid2 ) ){

// жоден з кандидатів не доступний

// вибір випадкового гену, який є відсутнім в дитини

int r = j = rand.Next( length );

// спершу йти вниз

while ( ( r < length ) && ( geneIsBusy[r] == true ) )

r++;

if ( r == length ){

// не знайдено, йти вгору

r = j - 1;

while ( geneIsBusy[r] == true ) // && ( r >= 0 )

r--;}

prev = (ushort) r;}

else{

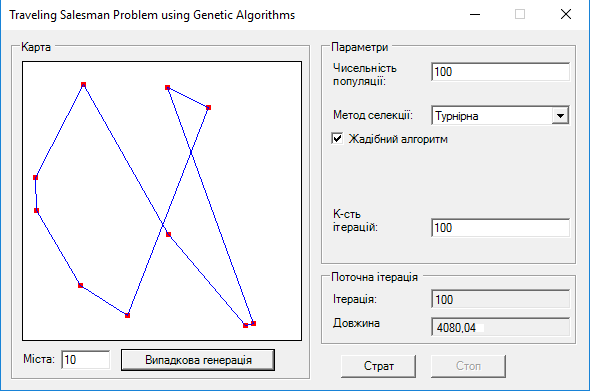
// один з кандидатів доступний

prev = ( valid1 ) ? next1 : next2;}

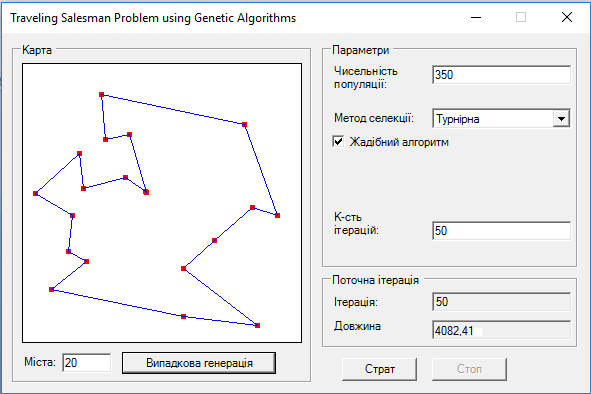
child[i] = prev;

geneIsBusy[prev] = true;}}}}

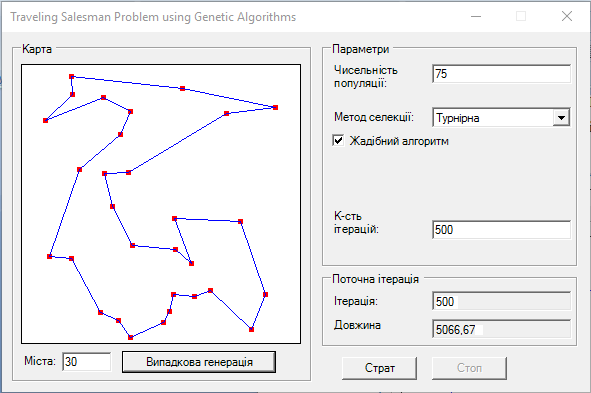
**Результат виконання**

****

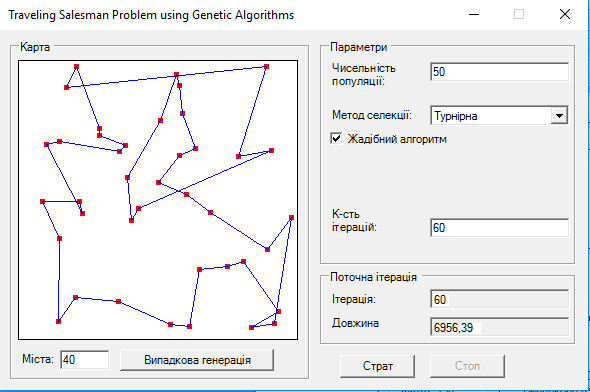
**Рис.1 Результат виконання програми з параметрами: к-сть міст - 10, чисельність популяції - 100, кількість поколінь - 100**

****

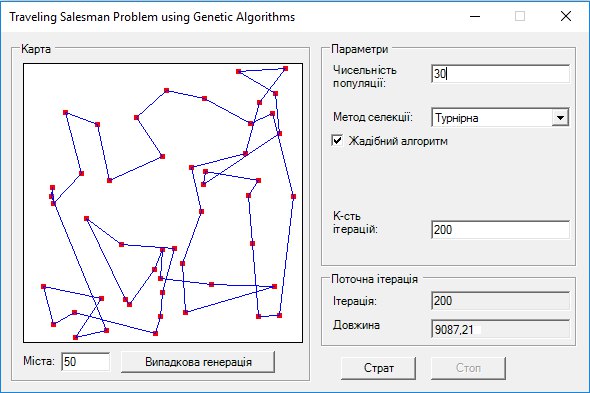
**Рис.2 Результат виконання програми з параметрами: к-сть міст - 20, чисельність популяції - 350, кількість поколінь - 50**

****

**Рис.3 Результат виконання програми з параметрами: к-сть міст - 30, чисельність популяції - 75, кількість поколінь - 500**

****

**Рис.4 Результат виконання програми з параметрами: к-сть міст - 40, чисельність популяції - 50, кількість поколінь - 60**



**Рис.5 Результат виконання програми з параметрами: к-сть міст - 50, чисельність популяції - 30, кількість поколінь - 200**

**Висновок**

Виконуючи лабораторну роботу я ознайомився з основними еволюційними операторами схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. Розробив за допомогою Microsoft Visual Studio 2013 програмне забезпечення для вирішення задачі комівояжера.