**Мета роботи**: Ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори, що використовуються при розв’язуванні задач комбінаторної оптимізації.

**Завдання:** Розробити на довільній мові програмування програмне забезпечення для вирішення задачі комівояжера.

В – 1

Використовуючи турнірну селекцію.

Результати виконання програми

При запуску для 10 міст з координатами:

[17 19; 2 19; 13 2; 5 11; 20 20; 3 20; 20 10; 16 2; 8 19; 16 20;]

Рішення:

|20, 20|16, 20|17, 19|8, 19|3, 20|2, 19|5, 11|13, 2|16, 2|20, 10|

При запуску для 20 міст з координатами) :

[4 5; 1 14; 19 3; 14 16; 18 10; 8 15; 9 11; 6 5; 15 0; 13 4; 13 5; 18 8; 15 5; 8 5; 11 7; 2 3; 14 5; 17 3; 14 5; 13 7]

Рішення:

|18, 8|15, 5|17, 3|19, 3|15, 0|13, 5|18, 10|14, 5|13, 4|13, 7|11, 7|6, 5|4, 5|2, 3|1, 14|8, 15|14, 16|9, 11|8, 5|14, 5|

При запуску для 30 міст з координатами:

[1 7; 12 7; 13 11; 4 5; 6 12; 12 9; 5 5; 20 13; 10 13; 17 5; 2 6; 13 4; 13 8; 17 16; 20 17; 19 10; 10 7; 19 1; 20 8; 8 12; 6 13; 17 19; 8 8; 2 9; 12 10; 18 8; 2 4; 13 0; 0 20; 15 17]

Рішення:

|17, 19|15, 17|12, 10|10, 13|12, 7|19, 10|18, 8|20, 8|13, 11|6, 12|0, 20|8, 12|6, 13|8, 8|2, 9|2, 6|4, 5|5, 5|2, 4|1, 7|13, 8|13, 4|10, 7|12, 9|19, 1|17, 5|13, 0|20, 13|20, 17|17, 16|

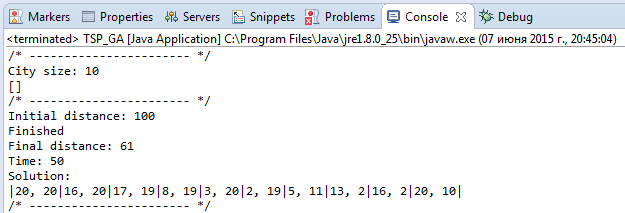


Рис.1. Результат програми при кількості міст 10 і популяції 50.

Таблиця порівняння залежності кількості міст і популяції

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість міст | 10 | | | 20 | | | 30 | | |
| Популяція | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| Час виконання,c | 0.3 | 0.9 | 1.0 | 0.15 | 0.3 | 0.9 | 0.1 | 0.4 | 0.9 |
| Мін. довжинна в Matlab(3 лаб) | 61.68 | 61.93 | 64.25 | 93.46 | 103.1 | 110.4 | 184.0 | 157.46 | 116.13 |
| Мінімальна довжина | 63 | 59 | 59 | 78 | 71 | 79 | 155 | 131 | 134 |

Код програми

**class TSP\_GA**

**int** citySize = 10;

**int**[] x = {17, 2, 13, 5, 20, 3, 20, 16, 8, 16};//new int[citySize];

**int**[] y = {19, 19, 2, 11, 20, 20, 10, 2, 19, 20};//new int[citySize];

Random random = **new** Random();

System.***out***.println("/\* ----------------------- \*/");

System.***out***.println("City size: " + citySize);

System.***out***.print("[");

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

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x, y, 20);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x, y, 100);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x, y, 200);

citySize = 20;

**int**[] x1 = {4 , 1, 19, 14, 18, 8, 9, 6, 15, 13, 13, 18, 15, 8, 11, 2, 14, 17, 14, 13};

**int**[] y1 = {5, 14, 3, 16, 10, 15, 11, 5, 0, 4, 5, 8, 5, 5, 7, 3, 5, 3, 5, 7};

System.***out***.println("/\* ----------------------- \*/");

System.***out***.println("City size: " + citySize);

System.***out***.print("[");

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

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x1, y1, 20);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x1, y1, 100);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x1, y1, 200);

System.***out***.println("/\* ----------------------- \*/");

**int**[] x2 = {1, 12, 13, 4, 6, 12, 5, 20, 10, 17, 2, 13, 13, 17, 20, 19, 10, 19, 20, 8, 6, 17, 8, 2, 12, 18, 2, 13, 0, 15};

**int**[] y2 = {7, 7, 11, 5, 12, 9, 5, 13, 13, 5, 6, 4, 8, 16, 17, 10, 7, 1, 8, 12, 13, 19, 8, 9, 10, 8, 4, 0, 20, 17};

citySize = 30;

System.***out***.println("/\* ----------------------- \*/");

System.***out***.println("City size: " + citySize);

System.***out***.print("[");

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

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x2, y2, 20);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x2, y2, 100);

System.***out***.println("/\* ----------------------- \*/");

**new** TSP\_GA(x2, y2, 200);

System.***out***.println("/\* ----------------------- \*/");

**class GA**

**public** GA(TourManager tourManager) {

**this**.tourManager = tourManager;

}

// розвивається населенням більше одного покоління

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

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

**int** elitismOffset = 0;

// кросовер населення

// Цикл розміром нового населення і створення осіб поточного населення

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

// Вибір батьків

Tour parent1 = tournamentSelection(pop);

Tour parent2 = tournamentSelection(pop);

// кросовер батьків

Tour child = crossover(parent1, parent2);

// добавити нащадка до нової популяції

newPopulation.saveTour(i, child);

}

// Провести мутацію нової популяції

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

mutate(newPopulation.getTour(i));

}

**return** newPopulation;

}

// двохточкове впорядковуюче

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

// створити новий прохід нащадка

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

**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;

}

// класичне інвертування

**private** **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);

}

}

**Висновки:** виконавши лабораторну роботу я вивчив еволюційні оператори, що використовуються при розв’язуванні задач комбінаторної оптимізації. Реалізував за допомогою мови програмування Java програмне забезпечення для вирішення задачі комівояжера з двохточкова мутація і рівномірне схрещування. В результаті програма коректно працює для кількість міст до 10, з більшою кількістю міст шлях комівояжера не оптимальний, але програма показує набагато кращі результати у порівнянні з реалізацією за допомогою пакету Matlab в 3 лабораторній роботі.