|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, САПР, СПК** | | **Тема** | **Оцінка:** | **Підпис:** |
| КНМ-14 | 3 | Комбінаторна оптимізація  за допомогою еволюційних  методів |  |  |
| Магеровський А.В. | |
|  | |
| Методи нечіткої логіки та  еволюційні алгоритми | | **Викладач:**  Кривий Р. З. | |

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

**Теоретичні відомості**

При використанні методів еволюційного пошуку для розв’язку задач комбінаторної оптимізації, як правило, застосовуються негомологічні числові хромосоми, тобто такі хромосоми, гени яких можуть приймати значення в заданому інтервалі. При цьому інтервал однаковий для всіх генів, але в хромосомі не може бути двох генів з однаковим значенням.

Комбінаторні задачі оперують із дискретними структурами або розміщенням об'єктів, незначні зміни яких часто викликають стрибкоподібну зміну показників якості (фітнесс- функції). Традиційні оператори еволюційні оператори, що генерують нових нащадків, не можуть бути застосовані при використанні негомологічних хромосом, оскільки внаслідок виконання таких операторів генеруються нащадки, що містять однакові гени і тому не можуть бути інтерпретовані при розв’язку комбінаторної задачі. Тому для розв’язку задач комбінаторної оптимізації були розроблені спеціальні генетичні оператори, що не створюють неприпустимих рішень.

**Завдання**

Розробити за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера. Параметри еволюційного методу обрати з таблиці 1 відповідно до варіанту.

|  |  |  |
| --- | --- | --- |
| № | Еволюційні оператори | |
| Схрещування | Мутація |
| 4 | Жадібне | Класичне інвертування |

**Хід роботи**

Для виконання завдання була використана функція ga пакету Matlab, і реалізовано власні функції мутації та схрещування, згідно з варіантом.

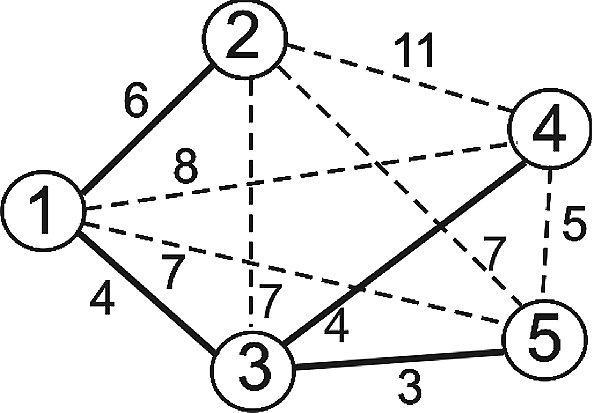


Рис. 1. Графічне представлення задачі комівояжера

Оператор схрещення:

function [xoverKids] = CrossoverFcn( parents, options, nvars, FitnessFcn, ...

unused,thisPopulation )

ret = zeros(length(parents)/2, nvars);

for i = 1:2:length(parents)-1

p1 = thisPopulation(parents(i), :);

p2 = thisPopulation(parents(i+1), :);

index = randi(nvars);

child = zeros(1,nvars);

point1\_1=0;

point1\_2=0;

point2\_1=0;

point2\_2=0;

child(1,1)=p1(index);

for j = 2:1:nvars

point1\_1=p1(index);

if index==5

index=0;

end;

point2\_1=findAppropriateCoord(p1,child,point1\_1,index);

indexOfElem=find(p2==point1\_1);

point1\_2=p2(indexOfElem);

if indexOfElem==5

indexOfElem=0;

end;

point2\_2=findAppropriateCoord(p2,child,point1\_2,indexOfElem);

dist1=findDistance(point1\_1,point2\_1);

dist2=findDistance(point1\_2,point2\_2);

if(dist1<=dist2)

child(1,j)=point2\_1;

index=find(p1==point2\_1);

else

child(1,j)=point2\_2;

index=find(p1==point2\_2);

end;

end;

ret((i+1)/2,:) = child;

end;

xoverKids = ret;

end

Оператор мутації:

function [ mutationChildren ] = MutationFcn( parents, options, nvars, ...

FitnessFcn, state, thisScore, thisPopulation )

mutant = thisPopulation(parents, :);

positions=[randi(nvars),randi(nvars)];

positions=sort(positions);

diff=positions(2)-positions(1);

avg=(positions(2)+positions(1))/2;

if(diff>1)

middle = round(avg);

for i=positions(1):1:positions(2)

if (i+1)==middle

break;

end;

temp=mutant(i+1);

mutant(i+1)=mutant(positions(2)-1);

mutant(positions(2)-1)=temp;

end;

end;

mutationChildren = mutant;

end

Точка запуску програми:

startPopulation = [

1, 3, 2, 4, 5;

2, 3, 4, 5, 1;

3, 1, 5, 4, 2;

4, 2, 1, 5, 3;

5, 1, 2, 3, 4

];

options = gaoptimset(...

'EliteCount', 0, ...

'PopulationSize', 5, ...

'InitialPopulation', startPopulation, ...

'MutationFcn', @MutationFcn, ...

'CrossoverFcn', @CrossoverFcn, ...

'TimeLimit', 3 ...

);

[x,fval,exitflag,output,population,scores] = ga(@optim\_function, 5, options);

disp('The best child:'); disp(x);

fprintf('f(x) = %d\n', fval);

disp('Last population:');

for i=1:1:5

for j=1:1:5

fprintf('\t%d', population(i,j));

end;

fprintf('\t=>\t%d\n', scores(i));

end;

Результат виконання

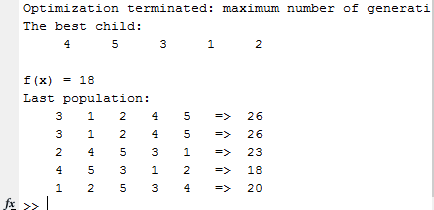


Рис.2.Результати виконання(прогін 1)

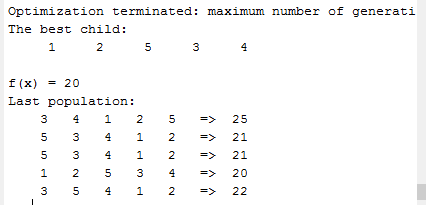


Рис.3.Результати виконання(прогін 2)

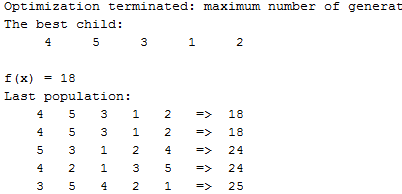


Рис.4.Результати виконання(прогін 3)

**Висновок**

На відміну від класичних методів розв’язання задачі комівояжера, використовуючи генетичні алгоритми, ми зразу отримуємо декілька оптимальних варіантів. Але такий підхід не гарантує, що результат є найоптимальніший.