*дата :*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНC-13 | Лаб.роб.3 | **Комбінаторна оптимізація за допомогою еволюційних методів** |  |  |
| Селезень А.Ю. | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р. З. | |

**Мета роботи:** ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. Розробити за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера.

**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.

* *Впорядковуючий оператор схрещування*

Впорядковуючий оператор схрещування (Order crossover, OX) був запропонований Д. Девісом у 1985 році для негомологічних числових хромосом. Схрещування може виконуватися по одній або по двох точках. Точки схрещування вибираються випадково. При одноточковому схрещуванні в хромосому першого нащадка копіюється хромосома першого батька, а потім гени нащадка, що розташовані правіше точки схрещування, перевпорядковуються у послідовність, що відповідає другому батькові. При цьому другий батько переглядається від початку до кінця, зліва направо, і елементи, яких не вистачає у нащадку, додаються, починаючи від точки схрещування, один по одному.

* Крок 1. Випадковим чином вибрати точку схрещування.
* Крок 2. Скопіювати в хромосому першого нащадка сегмент хромосоми першого батька, що розташований лівіше точки схрещування.
* Крок 3. Інші гени в нащадку копіюються із другого батька в упорядкованому вигляді зліва направо, крім елементів, які вже увійшли до нащадка.

Для створення другого нащадка застосовується аналогічний порядок дій.

* *Мутація обміну*

Мутація обміну використовується для бінарних і числових негомологічних хромосом. При класичній мутації обміну в хромосомі випадковим чином вибираються два гени, які міняються місцями.

* Крок 1. Створити хромосому нащадка як копію батьківської хромосоми

H = {h1,h2,...,hL}.

* Крок 2. Вибрати два числа y1 і y2 випадковим чином із множини Y = {0, 1, 2, .... L+1}, причому y1≠y2.
* Крок 3. Сформувати нову хромосому H шляхом обміну елементів, розташованих на позиціях y1 і y2.

Таким чином, після застосування класичної мутації обміну одержуємо

хромосому H'={h1­, h2, …,hy1-1, hy2, hy1+1, …, hy2-1, hy1, hy2+1, …, hL}.

**Індивідуальне завдання.**

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

****

**Фрагменти коду програми для схрещування та мутації**

% схрещування:

ii=roulette\_wheel\_indexes(ps,p); % гени з номерами міст ii будуть використані в схрещуванні

Gc=G(ii,:); % гени для схрещування

Gch=zeros(ps,nn); % діти

for prc=1:(ps/2) % підрахунок пар

i1=1+2\*(prc-1);

i2=2+2\*(prc-1);

g1=Gc(i1,:);

g2=Gc(i2,:);

cp=ceil((nn-1)\*rand); % точка схрещування, випадкове число в інтервалі [1; nn-1]

% двоє дітей:

g1ch=insert\_begining(g1,g2,cp);

g2ch=insert\_begining(g2,g1,cp);

Gch(i1,:)=g1ch;

Gch(i2,:)=g2ch;

end

G=Gch; % нові діти

% мутація обміну двох випадкових міст:

for psc=1:ps

if rand<pm

rnp=ceil(nn\*rand); % випадкове число міст для перестановки

rpnn=randperm(nn);

ctp=rpnn(1:rnp); %вибір випадкових міст для перестановки

Gt=G(psc,ctp); % отримати міста зі списку

Gt=Gt(randperm(rnp)); % переставити міста

G(psc,ctp)=Gt; % % повернути міста назад

end

end

% мутація обміну 2 частин шляху:

for psc=1:ps

if rand<pm2

cp=1+ceil((nn-3)\*rand); % діапазон [2 nn-2]

G(psc,:)=[G(psc,cp+1:nn) G(psc,1:cp)];

end

end

% мутація обміну випадкової частини шляху:

for psc=1:ps

if rand<pmf

n1=ceil(nn\*rand);

n2=ceil(nn\*rand);

G(psc,n1:n2)=fliplr(G(psc,n1:n2));

end

end

G(1,:)=Gb;

end

**Результат виконання**

Імовірність мутації обміну 2 випадкових міст на шляху = 0.01. Імовірність мутації обміну 2 частин шляху = 0.02. Імовірність мутації обміну випадкової частини шляху = 0.08.

**№1.** Кількість міст = 15. Чисельність популяції = 30. Кількість поколінь = 50.

Найкраща відстань між містами = 42.534

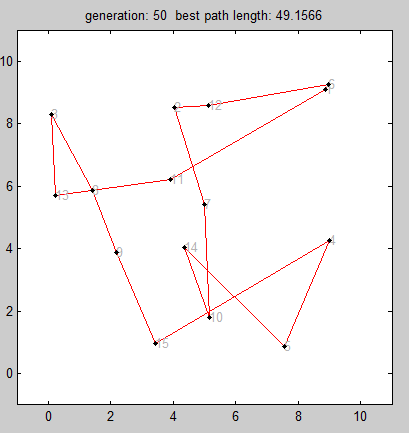


Рис.1. Результат виконання програми №1.

**№2.** Кількість міст = 65. Чисельність популяції = 200. Кількість поколінь = 110.

Найкраща відстань між містами = 251.4095

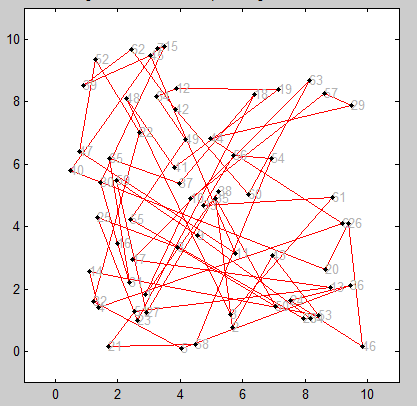


Рис.2. Результат виконання програми №2.

**№3.** Кількість міст = 35. Чисельність популяції = 450. Кількість поколінь = 80.

Найкраща відстань між містами = 120.6221

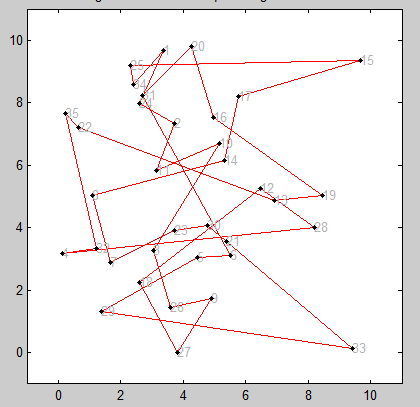


Рис.3. Результат виконання програми №3.

**№4.** Кількість міст = 80. Чисельність популяції = 50. Кількість поколінь = 300.

Найкраща відстань між містами = 262.151

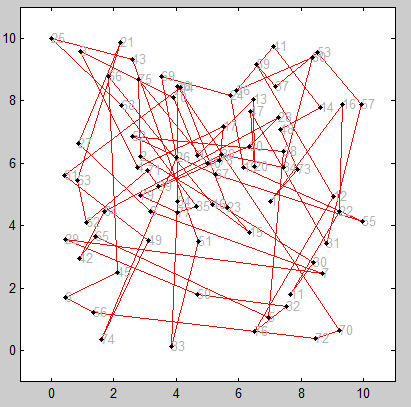


Рис.4. Результат виконання програми №4.

**Висновок**

Виконуючи лабораторну роботу, я ознайомився з основними еволюційними операторами схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. Розробив за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера. Здійснив кількаразовий запуск програми із різними параметрами популяції, к-стю міст та к-стю поколінь.