Прізвище: Березка

Ім’я: Тарас

Група: КНМ-14

Дата прийняття роботи

у системі Git:

Дисципліна: Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні

Перевірив: Кривий Р.З.

**Звіт до лабораторної роботи № 3**

**«Комбінаторна оптимізація за допомогою еволюційних**

**методів»**

**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.

**КОРОТКІ 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 одноточечному 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.

Мутація обміну

Мутація обміну використовується для бінарних і числових

негомологічних хромосом.

1. 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.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Розробити програмне забезпечення для розв’язку задачі комівожера.

|  |  |  |
| --- | --- | --- |
| **№ варіанту** | **Еволюційні оператори** | |
| **Схрещування** | **Мутація** |
| **1** | одноточечне впорядковуюче | класична мутація обміну |

**Фрагмент програмного коду:**

kst=40; % кількість міст

plot\_size=10; % розмір області asx x asz

pop\_kst=300; % чисельність популяції

pokol\_kst=5500; % Кількість поколінь

pm=0.01; % імовірність мутації двох міст

pm2=0.02; % імовірність мутації двох частин шляху

pmf=0.08; % імовірність мутації випадкової частини шляху

graph=plot\_size\*rand(2,kst); % випадкове розподілення міст

vid\_mat=zeros(kst,kst); % матриця відстаней

for n1=1:kst-1

r1=graph(:,n1);

for n2=n1+1:kst

r2=graph(:,n2);

dr=r1-r2;

dr2=dr'\*dr;

drl=sqrt(dr2);

vid\_mat(n1,n2)=drl;

vid\_mat(n2,n1)=drl;

end

end

% Почати з випадкових замкнених шляхів:

G=zeros(pop\_kst,kst);

for psc=1:pop\_kst

G(psc,:)=randperm(kst);

end

plot(1)

% вивід найкращого шляху:

hpb=plot(NaN,NaN,'r-');

ht=title(' ');

hold on;

% вивід ділянок вузла

for n=1:kst

text(graph(1,n),graph(2,n),num2str(n));

end

plot(graph(1,:),graph(2,:),'k.'); % виводити міста як чорні точки

axis equal;

%встановлення розміру графіка

xlim([-0.1\*plot\_size 1.1\*plot\_size]);

ylim([-0.1\*plot\_size 1.1\*plot\_size]);

pthd=zeros(pop\_kst,1); %довжина шляху

p=zeros(pop\_kst,1); % імовірності

%цикл поколінь

for gc=1:pokol\_kst

% знайти довжину шляху:

for psc=1:pop\_kst

Gt=G(psc,:);

pt=0; % підсумовування довжини шляху

for nc=1:kst-1

pt=pt+vid\_mat(Gt(nc),Gt(nc+1));

end

% останній і перший:

pt=pt+vid\_mat(Gt(kst),Gt(1));

pthd(psc)=pt;

end

ipthd=1./pthd; % зворотня довжина шляху

p=ipthd/sum(ipthd); % імовірності

[mbp bp]=max(p);

Gb=G(bp,:); % найкращий шлях

% Оновлення фігури

if mod(gc,5)==0

set(hpb,'Xdata',[graph(1,Gb) graph(1,Gb(1))],'YData',[graph(2,Gb) graph(2,Gb(1))]);

set(ht,'string',['Покоління: ' num2str(gc) ' Найкраща довжина шляху: ' num2str(pthd(bp))]);

drawnow;

end

% схрещування:

ii=roulette\_wheel\_indexes(pop\_kst,p); % іі - гени номери генів міст, що будуть використані

Gc=G(ii,:); % гени для схрещування

Gch=zeros(pop\_kst,kst); % діти

for prc=1:(pop\_kst/2) % підрахунок пар

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

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

g1=Gc(i1,:); % перший ген

g2=Gc(i2,:); % другий ген

cp=ceil((kst-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; % діти

% мутація обміну двох випадкових міст:

for psc=1:pop\_kst

if rand<pm

rnp=ceil(kst\*rand); % випадкове число міст для мутації

rpnn=randperm(kst);

ctp=rpnn(1:rnp); %вибрати rnp випадкових міст для мутації

Gt=G(psc,ctp); % отримати ці 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:pop\_kst

if rand<pm2

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

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

end

end

% мутація обміну випадкової частини шляху:

for psc=1:pop\_kst

if rand<pmf

n1=ceil(kst\*rand);

n2=ceil(kst\*rand);

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

end

end

G(1,:)=Gb; % елітарність

end

**Отримані результати:**

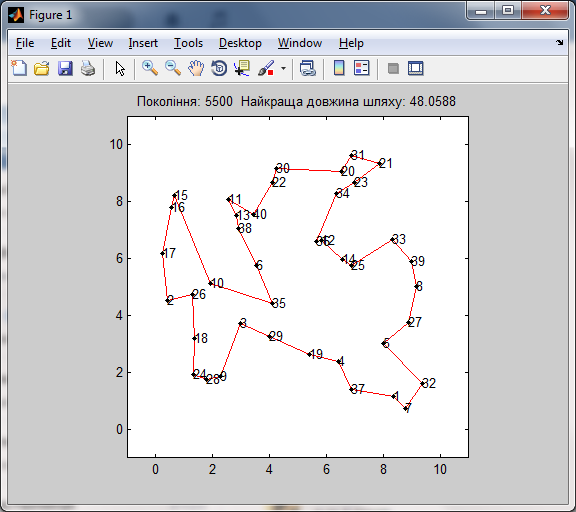


Рис 1.Результат при 40 містах та 5500 поколінь

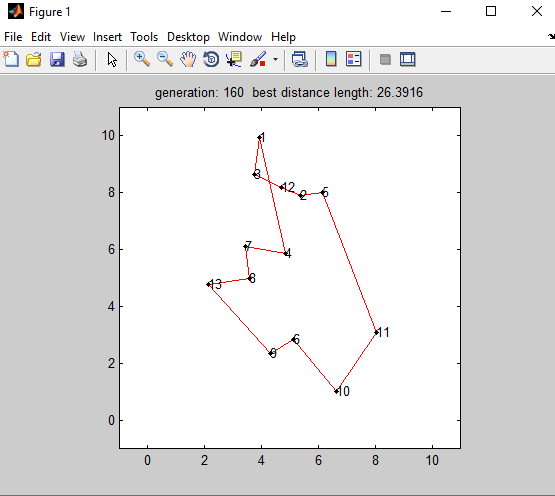


Рис 1.Результат при 40 містах та 5500 поколінь

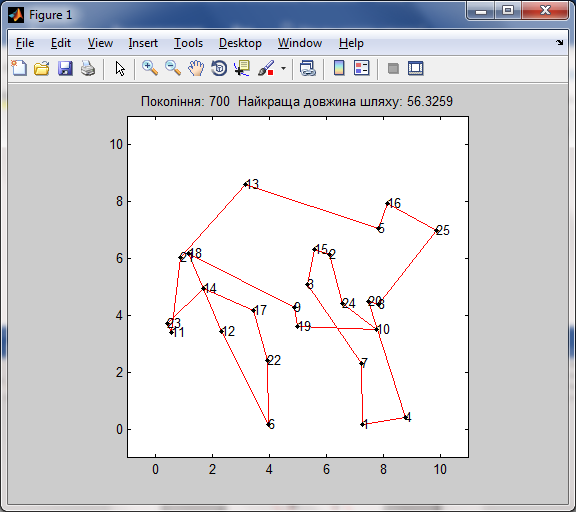


Рис 2.Результат при 25 містах та 700 поколінь

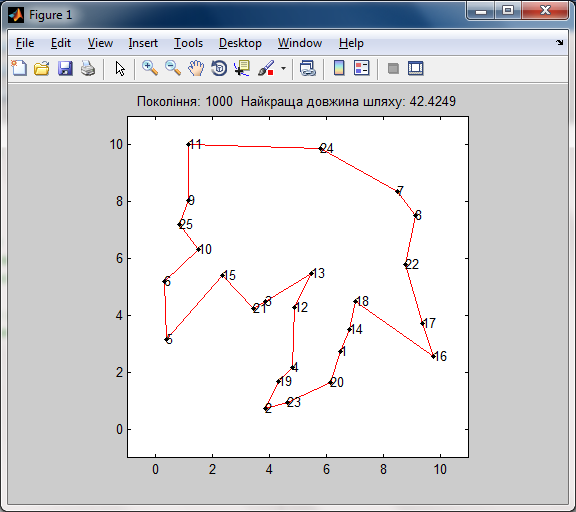
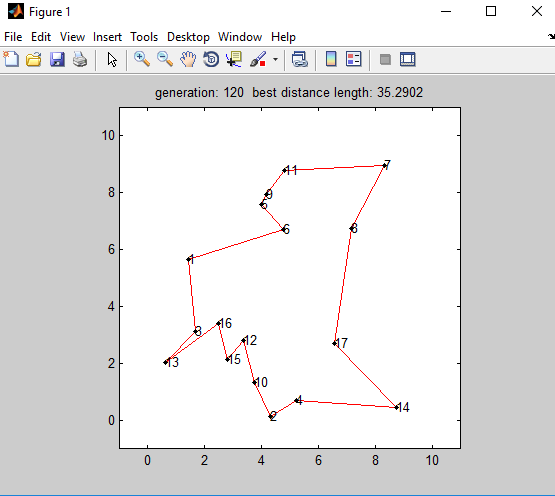


Рис 3.Результат при 25 містах та 1000 поколінь



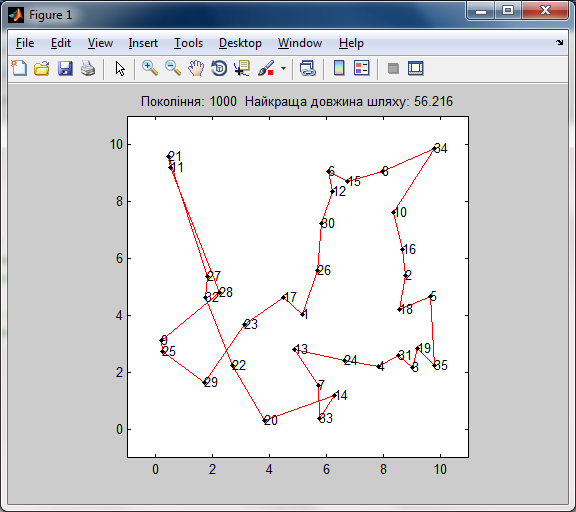


Рис 4.Результат при 35 містах та 1000 поколінь

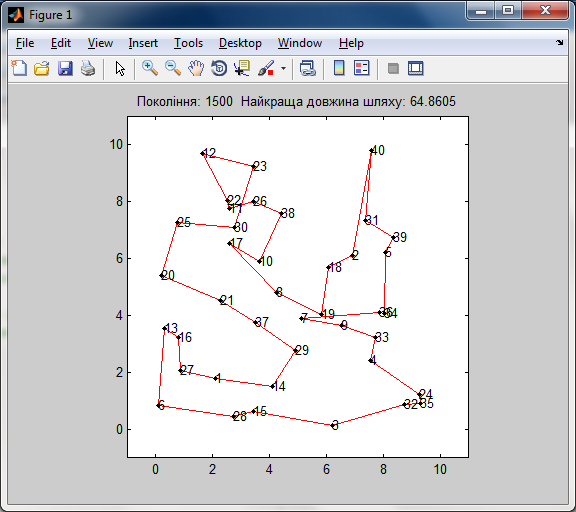


Рис 5.Результат при 40 містах та 1500 поколінь

**Висновок:** під час виконання лабораторної роботи ознайомився з основними теоретичними відомостями щодо генетичних алгоритмів, вивчив еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації та розробив за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера.