|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКс-11 | 3 | КОМБІНАТОРНА  ОПТИМІЗАЦІЯ ЗА ДОПОМОГОЮ ЕВОЛЮЦІЙНИХ  МЕТОДІВ |  |  |
| Якимець А.Р. | |
| № залікової: | |
| 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.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**Варіант -15(5)**

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





**Порядок виконання роботи**

1. Лістинг програми.

**traveling\_salesman\_problem.m**

cities = 40;%amount of cities

locations = zeros(cities,2);

mas = zeros(cities,2);

x=cell(1,1);

for n = 1: cities%generates city locations

locations(n,1) = randi([0 20],1,1);

mas(1,n) = locations(n,1);

locations(n,2) = randi([0 20],1,1);

end

x{1}=mas(1,:);

plot(locations(:,1),locations(:,2),'bo')

distances = zeros(cities);

for count1=1:cities,%calculate distance between cities

for count2=1:count1,

x1 = locations(count1,1);

y1 = locations(count1,2);

x2 = locations(count2,1);

y2 = locations(count2,2);

distances(count1,count2)=sqrt((x1-x2)^2+(y1-y2)^2);

distances(count2,count1)=distances(count1,count2);

end;

end;

x;

FitnessFcn = @(x) traveling\_fitness(x,distances);

my\_plot = @(options,state,flag) traveling\_s\_plot(options, ...

state,flag,locations);

timeBegin = clock;

options = gaoptimset('CreationFcn',@create\_permutations, ...

'CrossoverFcn',@crossover\_two\_point\_order, ...

'MutationFcn',@ivert\_with\_landslip, ...

'PlotFcn', my\_plot, ...

'Generations',500,'PopulationSize',100);

[x,fval,reason,output] = ga(FitnessFcn,cities,options)

timeEnd = clock;

time=timeBegin-timeEnd %time for optimization

displayEndOfDemoMessage(mfilename)

**create\_permutations.m**

function pop = create\_permutations(NVARS,FitnessFcn,options)

%function for creating population

totalPopulationSize = sum(options.PopulationSize);

n = NVARS;

pop = cell(totalPopulationSize,1);

for i = 1:totalPopulationSize

pop{i} = randperm(n);

end

**crossover\_two\_point\_order.m**

function xoverKids = crossover\_two\_point\_order(parents,options,NVARS, ...

FitnessFcn,thisScore,thisPopulation)

%two point order crossover

nKids = length(parents)/2;

xoverKids = cell(nKids,1); % Normally zeros(nKids,NVARS);

index = 1;

for i=1:nKids

%parent selection

parent1 = thisPopulation{parents(index)};

if(parents(index) == length(thisPopulation))

parent2 = thisPopulation{parents(index)-1};

else

parent2 = thisPopulation{parents(index)+1};

end

%generating 2 crossover points

p1 = randi([1 length(parent1)],1,1);

p2 = randi([p1 length(parent1)],1,1);

child = parent1;

child(p1:p2) = 0;

n = 0;

%inserting into son from father between

%2 points starting on the left

for j = 1:length(parent2)

if all(child ~= parent2(j))

child(p1+n) = parent2(j);

n = n+1;

end

end

xoverKids{i} = child;

index = index + 2;

end

**ivert\_with\_landslip.m**

function mutationChildren = ivert\_with\_landslip(parents ,options,NVARS, ...

FitnessFcn, state, thisScore,thisPopulation,mutationRate)

%invertion with move

mutationChildren = cell(length(parents),1);

for i=1:length(parents)

parent = thisPopulation{parents(i)};

p = ceil(length(parent) \* rand(1,2));%generating 2 vectors,

%1st is location of gene to be moved onto 2nd location

child = parent;

%cyclic movement to left

if(p(2)>p(1))

k1 = parent(p(2));

for j = (p(2)+1): length(parent)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

for j = 1: p(1)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

child(p(2)) = k1;

end

if(p(2)<p(1))

k1 = parent(p(1));

for j = p(2) : p(1)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

end

child(p(2)) = parent(p(1));

mutationChildren{i} = child;

end

end

**traveling\_fitness.m**

function scores = traveling\_fitness(x,distances)

%function for length of path

scores = zeros(size(x,1),1);

for j = 1:size(x,1)

p = x{j};

f = distances(p(end),p(1));

for i = 2:length(p)

f = f + distances(p(i-1),p(i));

end

scores(j) = f;

end

end

**traveling\_s\_plot.m**

function state = traveling\_s\_plot(options,state,flag,locations)

%graphics

[unused,i] = min(state.Score);

genotype = state.Population{i};

plot(locations(:,1),locations(:,2),'bo');

hold on;

plot(locations(genotype,1),locations(genotype,2));

hold off

end

1. Результат виконання завдання

Програма запускалася 9 раз з різною кількістю міст і популяцією. Для порівняння було вибрано кількість міст 10, 20, 40, а популяцію 20, 100,200.

Координати для 10 міст(рис. 1):

[12,2;5,3;15,15;17,3;8,16;4,13;6,14;6,19;5,14;2,9;]

Координати для 20 міст(рис. 2):

[7,20;13,8;4,13;5,18;12,15;8,15;6,7;13,17;1,13;11,6;16,3;20,12;16,12;6,19;13,5;11,10;13,12;7,3;16,3;19,2;]

Координати для 40 міст(рис. 3):

[5,15;11,3;0,14;13,3;14,11;3,5;3,10;9,4;9,8;7,8;3,15;2,20;13,1;8,10;13,7;19,12;4,0;5,20;11,16;19,20;13,14;11,3;9,9;10,17;8,11;18,9;11,19;11,6;5,7;9,18;9,0;11,3;20,3;13,19;10,11;8,6;0,2;20,19;14,9;17,2;]

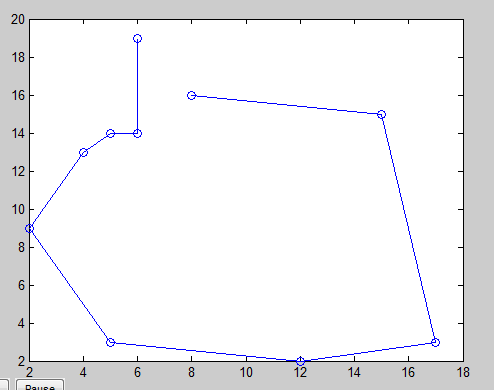


Рис.1. Рішення задачі комівояжера для 10 міст.

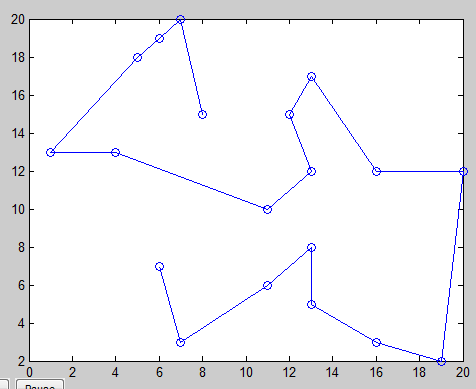


Рис.2. Рішення задачі комівояжера для 20 міст.

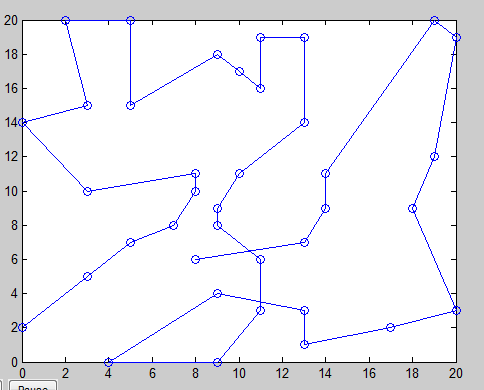


Рис.3. Рішення задачі комівояжера для 40 міст.

Таблиця порівняння кількості міст і популяції

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість міст | 10 | | | 20 | | | 40 | | |
| Популяція | 20 | 100 | 200 | 20 | 100 | 200 | 20 | 100 | 200 |
| Час виконання,c | 1.43 | 1.51 | 1.93 | 1.81 | 2.48 | 3.29 | 2.21 | 5.2 | 7.86 |
| Довжина шляху | 53.6 | 53.6 | 53.6 | 91.38 | 87.22 | 80.76 | 247.93 | 180.23 | 145 |
| Кількість  ітерацій | 51 | 51 | 51 | 60 | 58 | 51 | 82 | 122 | 115 |
| Кільк. оцінок функції | 1040 | 5200 | 10400 | 1220 | 5900 | 10400 | 1660 | 12300 | 23200 |

**Висновки:** під час виконання лабораторної роботи, я ознайомився з основними принципами застосування еволюційних методів до задач комбінаторної оптимізації. Згідно з лабораторним завданням я написав програму за допомогою середовища Matlab, що дозволяє розв’язувати різні види задач комівояжера з застосування генетичних алгоритмів.

На основі отриманих результатів тестування розробленого генетичного алгоритму, що застосовувався для розв’язання різних видів задачі комівояжера, можна зробити такі висновки: ефективність, тобто швидкість та точність результату, залежить від популяції та кількості міст. При збільшені популяції зростає точність та тривалість обрахунків. При збільшені кількості міст збільшується тривалість обрахунків та погіршується результат. Таким чином, цей алгоритм є ефективний лише у випадках малої кількості міст та невеликому населені, наприклад 10 міст з популяцією 20-100.