Assignment 03 - Evolutionary Algorithm to solve the tsp problem

This script contains the code to perform an evolutionary algorithm to solve the travelling salesman problem

Initialize Parameters

```
clear;
maximumGenerations = 50000;
populationSize = 30;
crossoverRate = 0.9;
nSpecies = populationSize/3;
cities = importdata('cities.csv');
numberOfGenes = length(cities.data);
mutationRate = 1/numberOfGenes;
% Use mode that showed most promising during experiments: no speciation
% random mutation with two point crossover
isSpeciationActivated = false;
isCrossoverOnePoint = false;
isMutationRandom = true;
```

Compute Distances

For reducing computation resources the distances between cities will be computed once and stored in a matrix for later usage. This could also be done using the pdist function, but as for this is already working...

```
distances = zeros(length(cities.data), length(cities.data));
for x = 1:length(cities.data)
    for y = 1:length(cities.data)
        distances(x,y) = sqrt((cities.data(x,2) - cities.data(y,2))^2 +...
            (cities.data(x,3) - cities.data(y,3))^2);
    end
end
```

Initialize population

Each phenotype will represent one possible route visiting each given city exactly once except the start city, which will also be the end city. Therefore the genotype can be represented as a sequence of the visited cities indices. This can easily be initialized by using randperm.

```
population = zeros(populationSize, length(cities.data));
for index = 1:populationSize
    population(index,:) = randperm(length(cities.data));
end
```

Evolution Loop

Initialization of loop parameters

```
bestFitness = zeros(maximumGenerations,1);
```

```
medianFitness = zeros(maximumGenerations,1);
tic;
for generation = 1:maximumGenerations
    % Evaluate population by computing the distance over the whole route
    % starting and ending at the first city. The fitness of an individual
    % will be the negative distance so that smaller distances will result
    % in higher fitness.
    fitness = -1* computeRouteDistance(population, distances);

    % Store fitness information for analysis and elitism
    [bestFitness(generation), best_index] = max(fitness);
    medianFitness(generation) = median(fitness);
    bestIndividual = population(best_index,:);

    % Speciate population by using kMeans with amount of clusters equal to
    % the parameter nSpecies.
    if isSpeciationActivated
        fitness = speciatePopulation(population, fitness, nSpecies);
    end

    % Use Tournament Selection by randomly choosing to sets of competitors
    % from the population and pair them up. The fitter ones will be chosen
    % to be the mates.
    matesA = tournamentSelect(population, fitness, populationSize);
    matesB = tournamentSelect(population, fitness, populationSize);

    % Recombination will either be one point or two point. Depending on
    % crossover rate either mateA will be the new offspring or a
    % combination of mateA and mateB. For one point crossover the first
    % part of the offspring will equal mateA and the second part will
    % contain the remaining genes in order of appearance within mateB. For
    % two point crossover the offspring will equal mateA outside of the two
    % crossover points, the order of the remaining genes within those
    % points will be equal to their order of appearance within mateB.
    if isCrossoverOnePoint
        population = onePointCrossoverSet(matesA, matesB, crossoverRate);
    else
        population = twoPointCrossoverSet(matesA, matesB, crossoverRate);
    end

    % Mutation will either swap each gene with a random other gene
    % or with the next gene in line depending on the mutation rate.
    if isMutationRandom
        population = mutateRandomly(population, mutationRate);
    else
        population = mutateNeighbours(population, mutationRate);
    end

    % Elitism takes over the best individual without changes. This will
    % remove one created offspring which is no problem neither in
    % computation time nor in loss of a special individual.
    population(1,:) = bestIndividual;
end
toc;
```

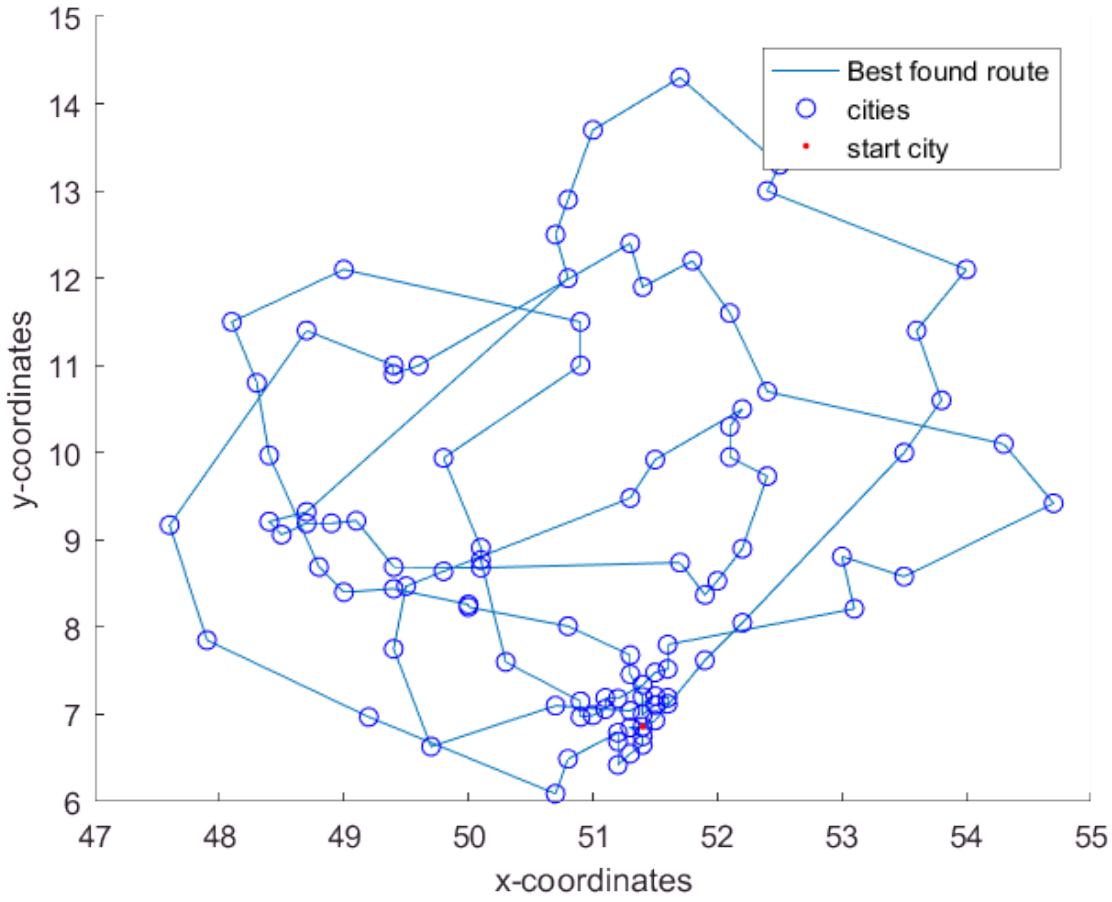
```
Elapsed time is 260.623074 seconds.
```

Show Results

Plot Route

Plot route (starting and ending at first city in list), all existing cities and mark start city individually.

```
figure(2);clf;hold on;
plot([cities.data(bestIndividual,2);cities.data(bestIndividual(1),2)],...
    [cities.data(bestIndividual,3);cities.data(bestIndividual(1),3)]);
plot(cities.data(:,2), cities.data(:,3),'bo');
plot(cities.data(bestIndividual(1),2), cities.data(bestIndividual(1),3),'r.');
xlabel('x-coordinates');
ylabel('y-coordinates');
legend('Best found route', 'cities', 'start city');
```

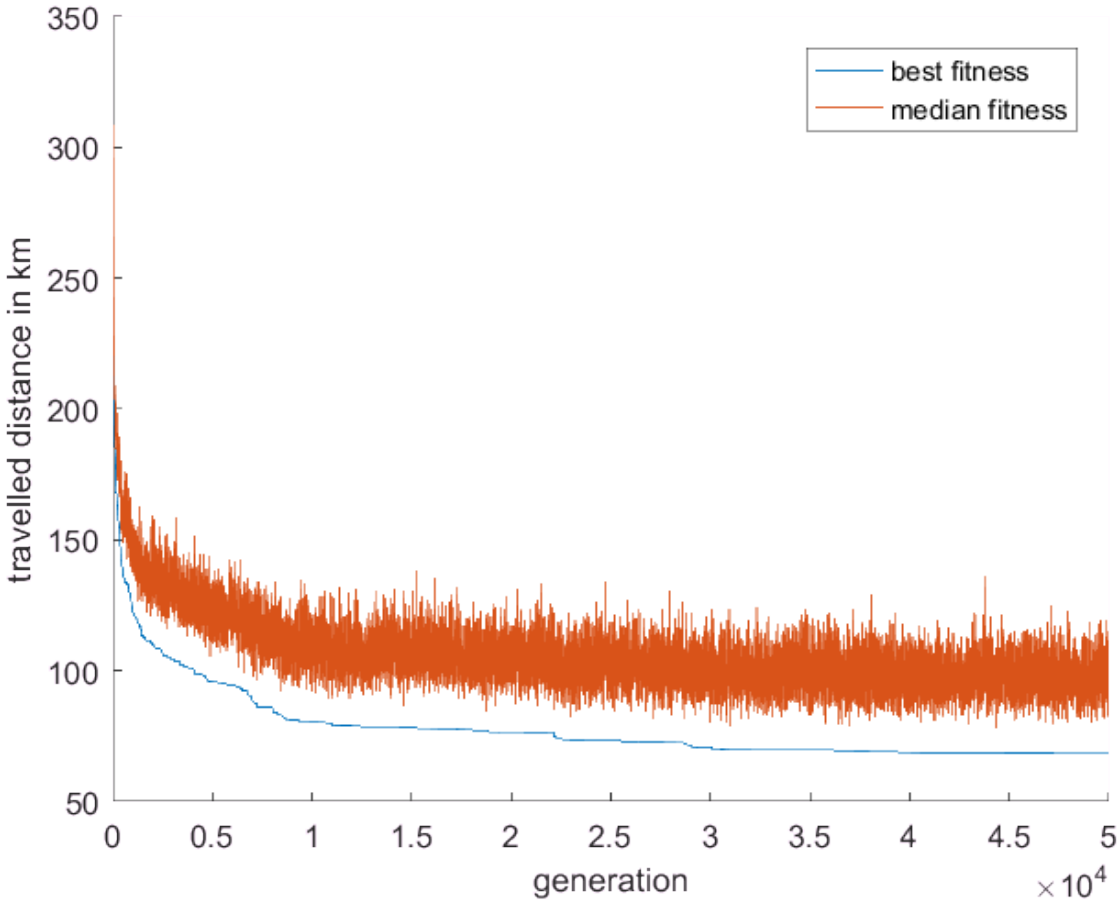


Plot Fitness Progression

Show fitness progression over time/generations

```
figure(3);clf;hold on;
plot(-1*bestFitness);
plot(-1*medianFitness);
ylabel('travelled distance in km');
xlabel('generation');
```

```
legend('best fitness','median fitness');
```



Display best Route

Show length and sequence of cities of best solution

```
lengthOfShortestRoute = -1* bestFitness(end);
disp(lengthOfShortestRoute)
```

```
    68.3550
```

```
BestRoute = cities.textdata(bestIndividual);
disp(BestRoute)
```

```
    'MÃ¼lheim'
    'Ratingen'
    'Duisburg'
    'Moers'
    'Krefeld'
    'MÃ¶nchengladbach'
    'Neuss'
    'DÃ¼sseldorf'
    'DÃ¼ren'
    'Aachen'
    'SaarbrÃ¼cken'
    'Freiburg'
    'Konstanz'
    'Ingolstadt'
```

'NÃ¼rnberg'
'FÃ¼rth'
'Erlangen'
'Leipzig'
'Halle'
'Dessau-RoÃŸlau'
'Magdeburg'
'Wolfsburg'
'Kiel'
'Flensburg'
'Bremerhaven'
'Bremen'
'Oldenburg'
'Hamm'
'LÃ¼nen'
'Dortmund'
'Witten'
'Wuppertal'
'Remscheid'
'Solingen'
'Leverkusen'
'KÃ¶ln'
'BergischGladbach'
'Koblenz'
'Offenbach'
'Hanau'
'WÃ¼rzburg'
'Erfurt'
'Jena'
'Regensburg'
'MÃ¼nchen'
'Augsburg'
'Ulm'
'Pforzheim'
'Karlsruhe'
'Ludwigshafen'
'Mainz'
'Wiesbaden'
'Siegen'
'Iserlohn'
'Hagen'
'Bochum'
'Essen'
'Gelsenkirchen'
'Marl'
'Recklinghausen'
'Herne'
'Velbert'
'Bonn'
'Trier'
'Kaiserslautern'
'Mannheim'
'Darmstadt'
'Kassel'
'GÃ¶ttingen'
'Braunschweig'
'Salzgitter'
'Hildesheim'
'Hannover'
'Minden'
'Bielefeld'
'GÃ¼tersloh'
'Paderborn'
'Frankfurt'
'Heidelberg'
'Heilbronn'
'Ludwigsburg'
'Stuttgart'

```
'TÃ¼bingen'
'Reutlingen'
'Esslingen'
'Gera'
'Zwickau'
'Chemnitz'
'Dresden'
'Cottbus'
'Berlin'
'Potsdam'
'Rostock'
'Schwerin'
'LÃ¼beck'
'Hamburg'
'OsnabrÃ¼ck'
'MÃ¼nster'
'Bottrop'
'Oberhausen'
```