**Assignment#5**

**Simulated Annealing Investigation**

Prepared For

Dr. Ken Ferens

University of Manitoba

Prepared by

Ali Mohammadzadeh

November 2017

In this reports all methods that I used to improve the code will be discussed. At the end of this report, the results will be shown and discussed.

**Change#1: Initialization parameters**

numCoolingLoops = 2000;

numEquilbriumLoops = 250;

pStart = 0.7; % Probability of accepting worse solution at the start

pEnd = 0.001;

**Change#2: Temperature Profile (Sigmoid function)**

xx = linspace(0,3,numCoolingLoops);

sigm = 1./(1+exp(xx));

tCurrent = tStart\*sigm(i) + tEnd;

**Change#3: Route Perturbation**

In my Simulated Annealing model, I used three different function for perturbation to increase diversity.

**First**: Swap, that is the change of position between two cities;

%% Swap

if Index==1

randIndex1 = randi(numCities);

alreadyChosen = true;

while alreadyChosen == true

randIndex2 = randi(numCities);

if randIndex2 ~= randIndex1

alreadyChosen = false;

end

end

dummy = theCityRoute(randIndex1);

theCityRoute(randIndex1) = theCityRoute(randIndex2);

theCityRoute(randIndex2) = dummy;

**Second**: Reversion changes the position of cities between two candidate cities.

For example, the two following cities (highlighted with red color) are our candidates, the reversion method will change the position of cities between these two as well.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| City | 15 | 26 | 35 | 5 | 42 |

%% Reversion

elseif Index==2

randIndex1 = randi(numCities);

alreadyChosen = true;

while alreadyChosen == true

randIndex2 = randi(numCities);

if randIndex2 ~= randIndex1

alreadyChosen = false;

end

end

if randIndex2<randIndex1

dummy1=randIndex1;

dummy2=randIndex2;

randIndex2=dummy1;

randIndex1=dummy2;

end

dummy=theCityRoute(randIndex1:randIndex2);

theCityRoute(randIndex1:randIndex2)=reversion(dummy);

**Third**: Insertion picks a city and puts it next to the second city.

For example, it pick city number 26 and put it next to the 5:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| City | 15 | 26 | 35 | 5 | 42 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 |
| City | 12 | 15 | 35 | 5 | 26 |

%% Insertion

elseif Index==3

randIndex1 = randi(numCities);

alreadyChosen = true;

while alreadyChosen == true

randIndex2 = randi(numCities);

if randIndex2 ~= randIndex1

alreadyChosen = false;

end

end

temp1=theCityRoute(randIndex1);

theCityRoute=insertion(temp1,randIndex1,randIndex2,theCityRoute);

end

I used “RouletteWheelSelection” method to choose a random route based on the probability I defined for each of approach (i.e. Swap, Reversion, Insertion):

Index=RouletteWheelSelection([0.05 0.95 0.95]); % Choose randomely between the following methods

**Change#4: Increasing Inner Loop Productivity**

To increase the explicitly of the algorithm (when some good solutions are found), the following command allows inner algorithm to search dipper for the solution. In another word, if it is noticed that inner algorithm is finding better results, the inner algorithm will continue for the better outcomes.

while D\_j-D\_b==0 || D\_j-D\_b<0

cityRoute\_j = perturbRoute1(numCities, cityRoute\_b);

D\_j = computeEUCDistance(numCities, cC, cityRoute\_j);

DeltaE = abs(D\_j-D\_b);

if (D\_j > D\_b) % objective function is worse

if (i==1 && j==1) DeltaE\_avg = DeltaE; end

p = exp(-DeltaE/(DeltaE\_avg \* tCurrent));

if (p > rand()) accept = true; else accept = false; end

else accept = true; % objective function is better

end

if (accept==true)

cityRoute\_b = cityRoute\_j;

D\_b = D\_j;

numAcceptedSolutions = numAcceptedSolutions + 1.0;

DeltaE\_avg = (DeltaE\_avg \* (numAcceptedSolutions-1.0) + ...

DeltaE) / numAcceptedSolutions;

end

The following command will lead algorithm to use a different function (i.e. perturbRoute1) for route Perturbation.

cityRoute\_j = perturbRoute1(numCities, cityRoute\_b);

It is good to notice that because in this case a smaller step is required, the probability of swap (which provides smaller changes) is increased.

Index=RouletteWheelSelection([0.95 0.05 0.05]);

**Change#5: Using Adoptive Cooling Strategies**

The fixed temperature reduction method is modified through the use of an adoptive temperature reduction strategies for outer loop. The following command will allow the algorithm to decrease the temperature only when the algorithm’s performance is positive. In another word, when the best cost generated by the inner loop is higher than the optimum cost obtained so far, it means that the exploration of algorithm couldn’t satisfy the solution and it is probably still in a local minimum. Therefore, the algorithm will increase the temperature to provide solutions with a more diverse exploring space, and in this way more solutions will be examined.

tempT=tCurrent;

tCurrent=(1+((D\_b-D\_o)/D\_b))\*tCurrent;

cityRoute\_o = cityRoute\_b; % Update optimal route at each cycle

D(i+1) = D\_b; %record the route distance for each temperature setting

D\_o = D\_b; % Update optimal distance

**Change#6: Different Termination Strategy for Outer Loop**

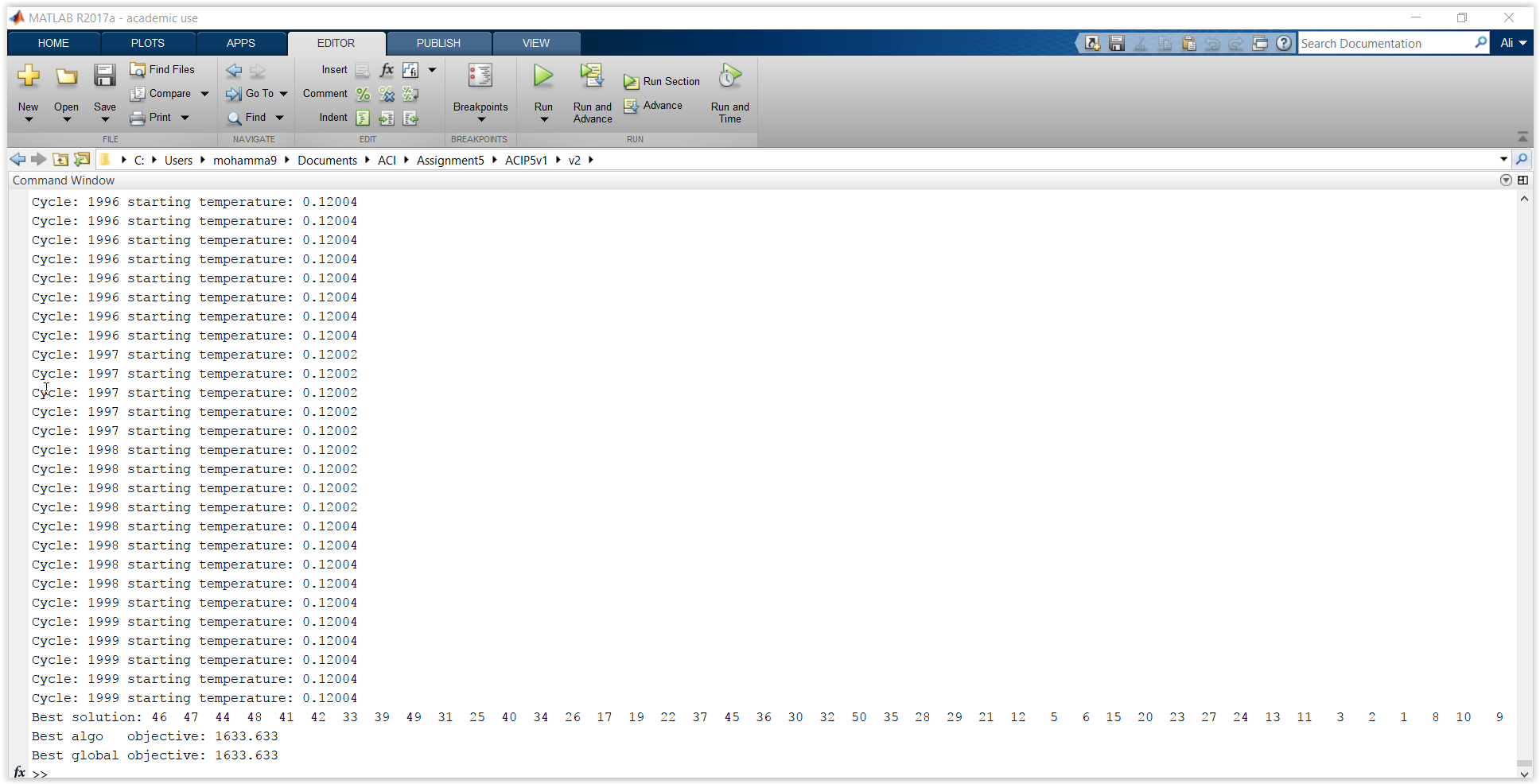
The following command allows algorithm to regulate the performance of the inner loop by comparing the current and previous temperature. It means that if the current temperature is higher than the previous one the algorithm will continue working without increasing the number of temperature. Therefore the algorithm will have more time to find better solutions.

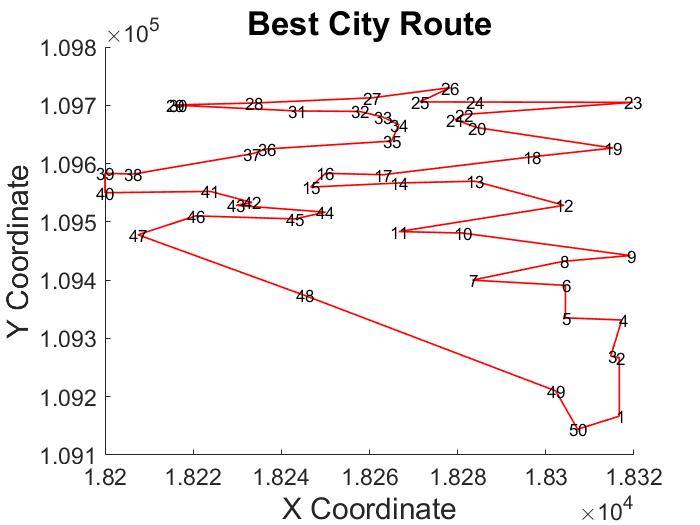
if tCurrent<tempT

i=i+1;

end

**Final Results:**

The following figure shows the final cost.

The following show the best solution (i.e. best route):

C:\Users\mohamma9\Documents\ACI\Assignment5\ACIP5v1\v2\Assignment#5\untitled.emf