CS168 Spring Assignment [number]

SUNet ID(s): johngold jsalloum
Name(s): John Gold Justin Salloum
Collaborators: None

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

Part 1

(a) There are 30! states in this markov chain. As maxiter tends to infinity and T = 0, we will not see all the states. This is because once we can get stuck in a local optima and never change much. When T is greater than 0, it's possible to see all the states, because the route can change even when it is not a lower distance.

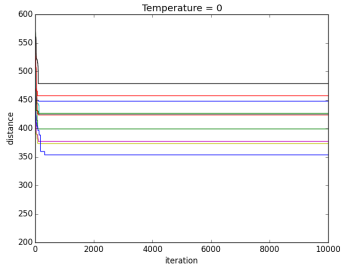
```
#p3.py
import math
import random
import copy
import numpy as np
import sys
import matplotlib.pyplot as plt


def main():
    t = int(sys.argv[1])
    #Contest("data/parksContest.csv", t)
    #Set to "D" for that 1D, also set 0 to 70 for part 2A
    Assignment("data/parks.csv", t, "B", 0)


#This runs the code for the Assignment, can be modified for parts B, D or
def Assignment(filename, t, part, c):
    parks_data = read_parks_csv(filename)
    num_trials = 10

    best_route_tracker_dict = {}
    for i in range(num_trials):
        best_route_tracker, best_route = MCMC(maxiter=10000, data=parks_d
        best_route_tracker_dict[i] = best_route_tracker
        best_dist = best_route_tracker[len(best_route_tracker) -1]
        print "best dist for trial ", i, " is : ", best_dist
    plot_figure(best_route_tracker_dict, t)
```

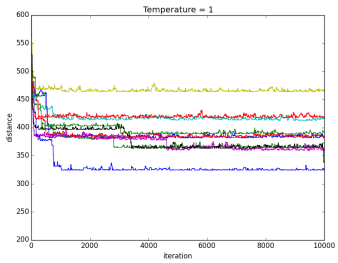
```
    #plot_best_route


'''
Runs our MonteCarlo MarkovChain Model
Input: number of max iterations
Input: dictionary - data {"A" : (1,3), "B" : (4, 8)...}
Input: number T temperature
Input: number c annealing factor (if 0, no annealing)
Input: string "B" to take 2 parks next to eachother in route
'''
def MCMC(maxiter=10000, data=None, T=10, c=0, part="B"):
    route_tracker = []
    route = list(data.keys())
    random.shuffle(route)
    best_route = route
    best_dist = route_dist(route, data)
    for i in xrange(maxiter):
        if(c > 0):
            T = c / math.sqrt(i+1)

        first, second = 0, 0
        if(part == "B"):
            first = random.randint(0, len(route)-1)
            second = (first + 1) % len(route)
        else: #Assumed to be D if not B
            first, second = random.sample(range(len(route)-1), 2)

        new_route = copy.deepcopy(route)

        #Switch the indicies, then compute new distance
        new_route[first], new_route[second] = new_route[second], new_rout
        new_dist = route_dist(new_route, data)
        delta_dist = new_dist - route_dist(route, data)
        if(delta_dist < 0 or (T > 0 and random.random() < np.exp(-delta_d
            route = new_route
        if(route_dist(route, data) < best_dist):
            best_route = route
            best_dist = route_dist(best_route, data)
        #print "Best distance is: ", best_dist
        route_tracker.append(route_dist(route, data))
    return route_tracker, best_route
```

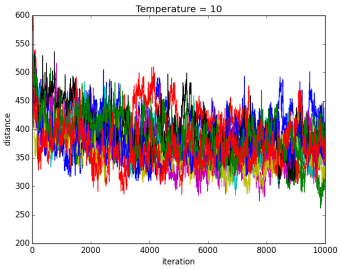
```
def plot_best_route(route, parks_data):
    x_coords = []
    y_coords = []
    #use the x coordinate for longitude and the y coordinate for latitude
    for park in route:
        x_coords.append(parks_data[park][0])
        y_coords.append(parks_data[park][1])
    x_coords.append(parks_data[route[0]][0])
    y_coords.append(parks_data[route[0]][1])

    plt.ylabel('Longitude')
    plt.xlabel('Latitude')
    plt.title('Best Route: ' + str(round(route_dist(route, parks_data), 2
    plt.plot(x_coords, y_coords, marker='o')#, linestyle='--', color='r')
    plt.show()


#Takes in a dict of y_coords
#Plot each of those as a line, with range(len(dict[0])) as x_coords
def plot_figure(best_route_tracker_dict, t):
    plt.ylabel('distance')
    plt.xlabel('iteration')
    title = "Temperature = " + str(t)

    plt.title(title)
    for key, value in best_route_tracker_dict.items():
        plt.plot(value)

    x1,x2,y1,y2 = plt.axis()
    plt.axis((x1,x2,100,600))

    plt.show()


'''
Reads in CSV of:
HEADER
park1, lat, long
park2, lat, long
```

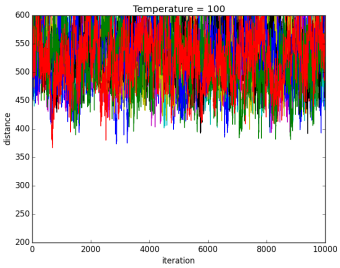
```
    into a dict of {park1 : (lat, long), park2: (lat, long)}
    '''
    def read_parks_csv(filename):
        park_data = {}
        f = open(filename)
        content = f.readlines()[1:] #Skip header
        for line in content:
            split_line = line.strip().split(",")
            park_data[split_line[0]] = (float(split_line[1]), float(split_lin
        return park_data

    #Calculates the distance between a route (list of parks)
    def route_dist(route, data):
        total_dist = 0
        cur_park = route[0]
        for i in range(1, len(route)):
            next_park = route[i]
            total_dist += park_dist(data, cur_park, next_park)
            cur_park = next_park
        #Route has to return to the start
        return (total_dist + park_dist(data, cur_park, route[0]))

    #Calculates the distance between two parks
    def park_dist(data, park1, park2):
        return dist(data[park1][0], data[park1][1], data[park2][0], data[park

    #Calculates distance given lats and longs
    def dist(lat1, long1, lat2, long2):
        return math.sqrt((lat1-lat2)**2 + (long1-long2)**2)

    if __name__ == "__main__":
        main()
```

(b) Plots





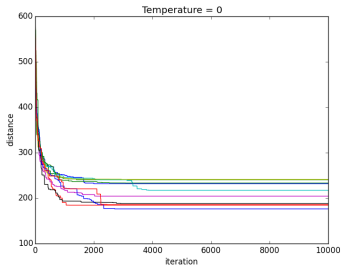


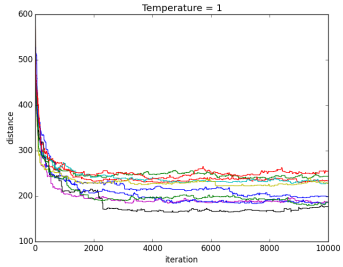


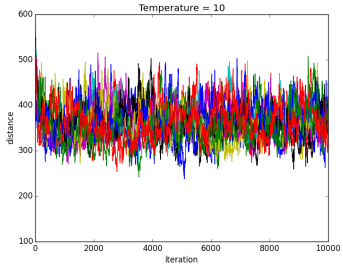
(c) When T = 0, the distances barely budge after the first 500 iterations. When T=1, there are slightly more decreases in distance through the iterations, but the lines are still generally straight. There is also a wider range of best distances compared to T=1. T=10 and T=100 share more in common with each other, namely lots of variation/changing of the route. There is a lot of mixing time when the T is increased. The best value of T is 10. This is where the global optimum is hit, and nearly all of the 10 trials find a distance lower than 350, the best score of any other temperature. This is definitely the sweet spot, a higher temperature leads to more states being explored, but when it is too high, there is an over-emphasis on exploration.

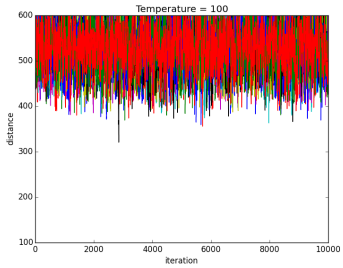
(d) The main difference is the algorithm for this part works better. T=1 becomes the sweet spot, and finds much lower distances than earlier. It is no longer limited by only picking two successsive parks, and can therefore check more of the different markov states. A wider variety of states are possible to find, especially at the beginning.

More Plots



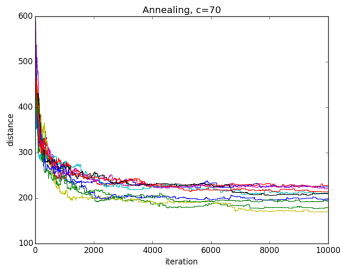






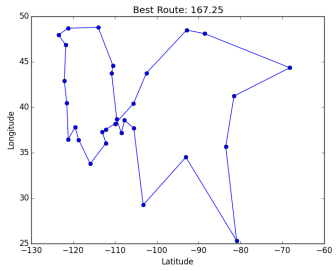
Part 2

(a) The code is intertwined with the code from earlier. Just set the 4th parameter in the call to Assignment() in main() from 0 to 70. Here are the plots:



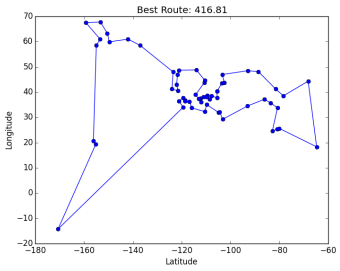
(b) Compared to our best graph from part D, (T=1), there is a slight improvement in the average min distance, though they both find similar global optima. However, simmulated annealing helps in a big way for the other values of T.

(c) This does look close to an optimal solution, as in no lines are crossing and the picture resembles a circle. There is at least one location that I would switch, although it would not involve switching the order of just two parks, it would require a permutation of 3 parks. This idea is taken farther in the bonus. More iterations would also help.



(d) Contest: We achieved a distance of 416.8, with the following route: ['Rocky Mountain', 'Great Sand Dunes', 'Black Canyon of the Gunnison', 'Mesa Verde', 'Arches', 'Canyonlands', 'Capitol Reef', 'Bryce Canyon', 'Grand Canyon', 'Zion', 'Great Basin', 'Grand Teton', 'Yellowstone', 'Glacier', 'North Cascades', 'Mount Rainier', 'Crater Lake', 'Lassen Volcanic', 'Redwood', 'Olympic', 'Glacier Bay', 'Wrangell St. Elias', 'Kenai Fjords', 'Denali', 'Gates of the Arctic', 'Kobuk Valley', 'Lake Clark', 'Katmai', 'Haleakal', 'Hawaii Volcanoes', 'American Samoa', 'Channel Islands', 'Pinnacles', 'Yosemite', 'Kings Canyon', 'Sequoia', 'Death Valley', 'Joshua Tree', 'Saguaro', 'Petrified Forest', 'Guadalupe Mountains', 'Carlsbad Caverns', 'Big Bend', 'Hot Springs', 'Mammoth Cave', 'Great Smoky Mountains', 'Congaree', 'Dry Tortugas', 'Everglades', 'Biscayne', 'Virgin Islands', 'Acadia', 'Shenandoah', 'Cuyahoga Valley', 'Isle Royale', 'Voyageurs', 'Theodore Roosevelt', 'Badlands', 'Wind Cave', ('Rocky Mountain')]

Here is another visualization:



Our strategy centered around using more than 2 random parks to swap. First we tried picking 3 at random, and then looking at the different permutations/route distances, then we tried 4 different random parks. We used simulated annealing with a c=200 after some hypterparameter tuning. We also increased maxiter to values between 100,000 and 500,000.