A2Q1:

State: every path which contains all cities exact once is a state

Neighbor relation: the neighbor of current state is to change any two of neighboring cities in current state(path)

Cost function: the total distance of current path (i.e. distance of a tour which begins at one city and visit each of other cities exact once)

**Average steps to reach local optimal:**

* 14 Cities

Instance 1:10 steps

Instance 2: 9 steps

Instance 3: 11

Instance 4: 11

Instance 5: 10

Instance 6: 10

Instance 7: 10

Instance 8: 9

Instance 9: 10

Instance 10:10

* 15 Cities

Instance 1:12 steps

Instance 2: 11 steps

Instance 3: 12

Instance 4: 11

Instance 5: 11

Instance 6: 10

Instance 7: 12

Instance 8: 9

Instance 9: 10

Instance 10:11

* 16 Cities

Instance 1:12 steps

Instance 2: 11 steps

Instance 3: 11

Instance 4: 12

Instance 5: 12

Instance 6: 13

Instance 7: 13

Instance 8: 11

Instance 9: 12

Instance 10:11

**average quality of the best solution (best solution found on given website):**

* 14 cities

Instance 1:1.1

Instance 2: 1.29

Instance 3: 1.08

Instance 4: 1.17

Instance 5: 1.14

Instance 6: 1.08

Instance 7: 1.04

Instance 8: 1.37

Instance 9: 1.28

Instance 10:1.09

* 15 cities

Instance 1:1.37

Instance 2: 1.29

Instance 3: 1.06

Instance 4: 1.1

Instance 5: 1.02

Instance 6: 1.25

Instance 7: 1.09

Instance 8: 1.56

Instance 9: 1.47

Instance 10:1.11

* 16 cities

Instance 1:1.21

Instance 2: 1.27

Instance 3: 1.02

Instance 4: 1.06

Instance 5: 1.09

Instance 6: 1.25

Instance 7: 1.03

Instance 8: 1.56

Instance 9: 1.37

Instance 10:1.04

**percentage of 100 repetitions where hill climbing found the same solution as the best solution found by the alternative algorithm**

* 14 cities

None for all instances

* 15 cities

None for all instances

* 16 cities

None for all instances

Conclusion: As increasing of city number, we can observe that all three average numbers above also become bigger. It is easy to analysis this trend since for more cities we have, there would be more possible neighbors for any states in my assumption of neighbor relationship. For example, in 14 cities instance, each state has 13 neighbors and in 15 cities there are 14 neighbors for each state. When we have more neighbors, in hill climbing algorithm, we would have more choices to pick next steps. When we compare our local optimal solution with the best solution (found by given website), ours are always worse than the best. Since we run too few times (<1,000,000 times) to find optimal solution, we could not be better than “the best” solution. Somehow, if we run enough times, maybe we would obtain a global optimal solution instead of a local optimal solution.

Choose 5-city from A1

The local optimal solution obtained by hill climb is [A, D, E, C, B] and cost is 158.525(round to 3 decimals)

Neighbor 1: [A, E, D, C, B] cost is 215.698

Neighbor 2: [D, A, E, C, B] cost is 162.347

Neighbor 3: [A, D, C, E, B] cost is 171.137617827

Neighbor 4: [A, D, E, B, C] cost is 170.576

Thus, this is strict local optimal solution

It supposed that the hill climb with sideway moves is worse then simple hill climb. I use 14-city instance\_1 as example. Simple hill climb is 474.910569718 but hill climb with sideways is 723.449826242. Obviously, the simple hill climb has better performance.

5.

14\_instance1

14\_instance2

15\_instance1

15-instacne2

16-instance1

16-instance2

For question 1, with my experiments, I would set 1000 restart times for that quality.

For question 2, I agree that there is the trade-off between the execution and solution quality. According to the data I provided in excel file, it is very obviously to note that with the growth of times of restart, the quality of solution is improved.

For question 3, I would choose 500 times for less city’s instances (such like 12 and 13 cities) since we could obtain good quality respect to execution time. 1000 times for more than 14 cities since from my experiments’ result in question 1. I would like to make sure the quality of solution in large number of city instance and we still could get “not bad” execution time namely less than 0.1s for one execution.

6.

14\_instance1

14\_instance2

15\_instance1

15\_instance2

16\_instance1

16\_instance2

For question 2, I think there exists the trade-off between the quality of solution and execution time. I use three schedules: linear decrease, exponential decrease and division decrease. Although they have slightly different in execution time, we still can find out that when we have more time to run the code, we would have better solution.

For question 3, I will use exponential decrease as my schedule since I could get the best quality of solution and acceptable execution time from my data result.

7.

For simple hill climb, cost is 1589.0784130922684

For hill climb with random start (start times = 100) cost is 1522.719212496732

For simulated annealing, cost is 1512.920436475001

Thus, obviously, we should use simulated annealing as our the best algorithm