Assignmnent 2

Tareef Dedhar - 20621325
CS 486 - AI

October 26, 2018

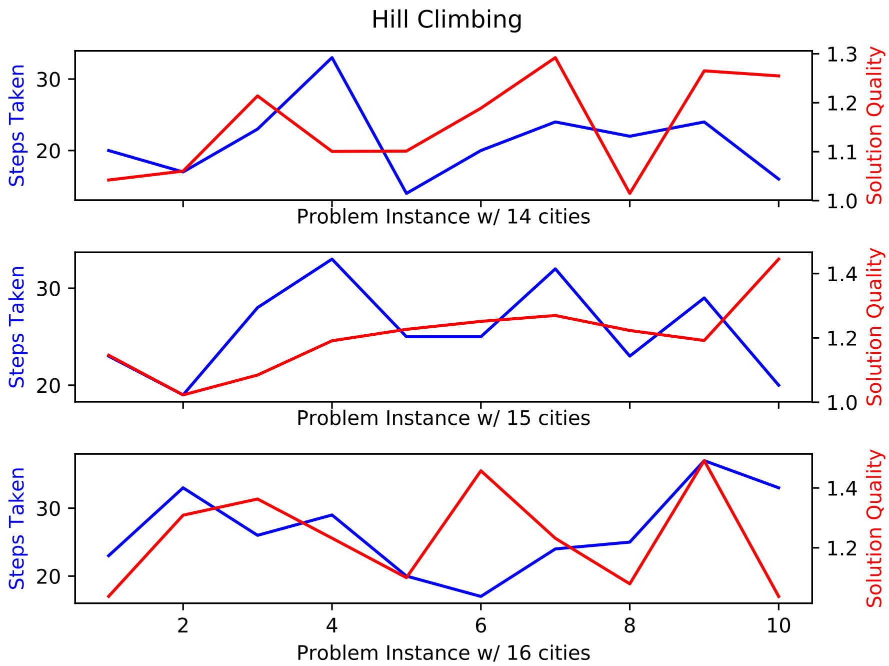
Question 1.1. Formulate the TSP problem as a local search problem. Include the definitions of a state, the neighbor relation, and the cost function.

Solution.

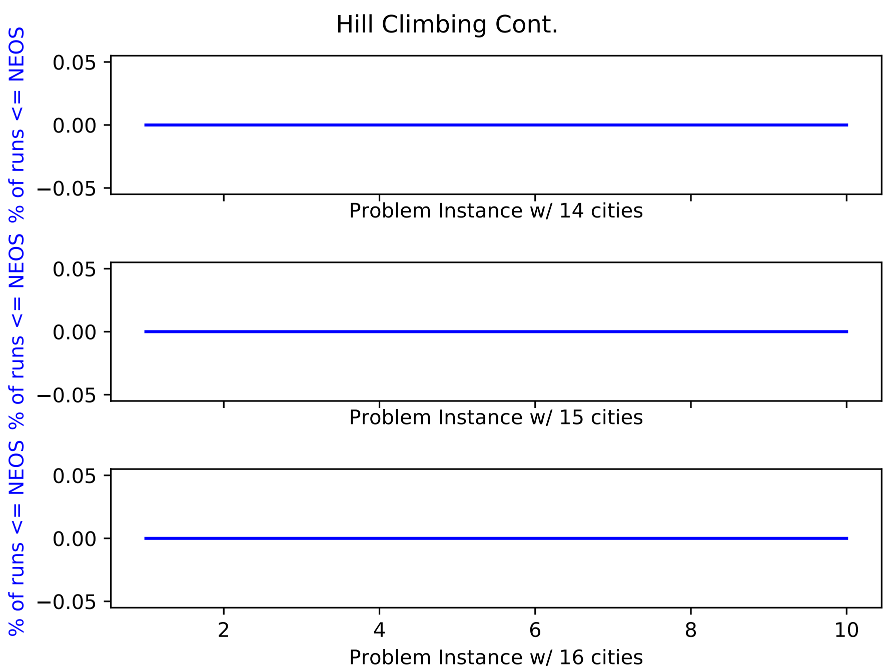
1. A state is defined as a sequence of all the cities, starting at the first and ending at it as well with no other repetition. This represents a tour the salesperson can take.
2. The neighbour relation is defined as follows: a neighbour of a state is a state in which 2 cities in the order (not the first or last, of course) have been swapped.
3. The cost/value function is calculated as the total distance travelled in the tour, so the Euclidean distance from the 1st to 2nd city + the 2nd to the 3rd, etc.

Question 1.2. Consider the 14, 15, 16-city instances. Solve the problem instances using hill climbing (with no sideways moves, no tabu list and no random restarts) and report your findings. For each number of cities and each problem instance, run hill climbing for 100 times and report the following averages (where the averages is taken over the 100 repetitions of the algorithm) The average number of steps it takes for hill climbing to reach a local optimum The average quality of the best solution The percentage of 100 repetitions where hill climbing found the same solution as the best solution found by the alternative algorithm Next, for each number of cities, calculate the average of each of the three numbers above over the ten instances. Report the three averages for each number of cities. How do the three average numbers change as the number of cities increases? Describe and discuss any trend that you can observe in no more than one paragraph.

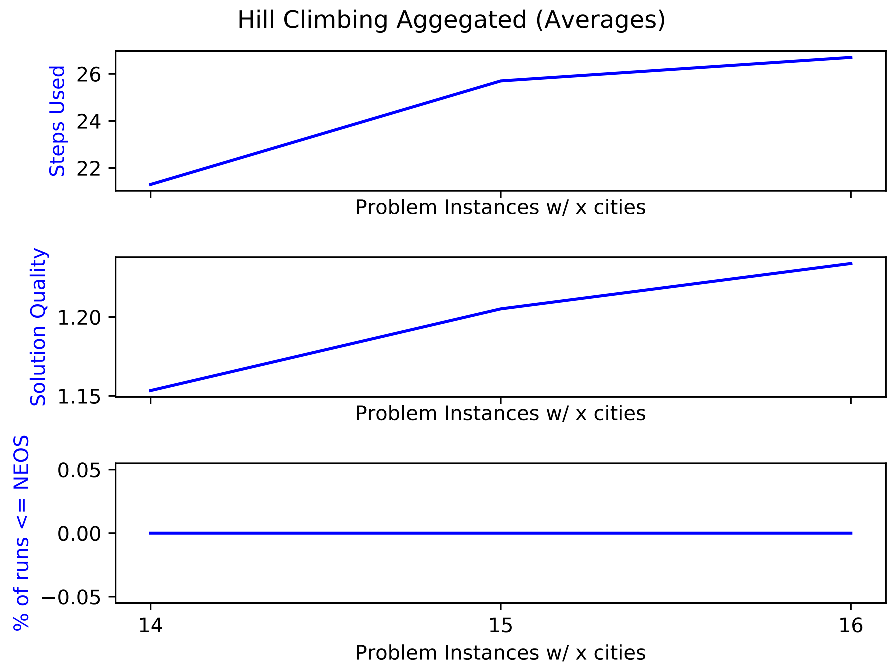
Solution.



Hill Climbing
Steps Taken
Solution Quality
Problem Instance w/ 14 cities
Problem Instance w/ 15 cities
Problem Instance w/ 16 cities



Hill Climbing Cont.
% of runs <= NEOS
Problem Instance w/ 14 cities
Problem Instance w/ 15 cities
Problem Instance w/ 16 cities



Regarding the steps taken, this seems to scale very slowly/minimally, as there was an average increase of 2.5 steps per extra city. Of course, our sample is rather small, but it seems that the impact of adding stops isn't overly high regarding iterations in this algorithm. As for solution quality, it definitely appears to decrease linearly as complexity increases. By decrease, we mean the hill climbing solution becomes worse relative to the reference (ratio increases). This indicates that we need a smarter algorithm when we have a larger search space and more potential local optima to get stuck in. As for the percentage of run equal or better than the reference, we had none, in any of the runs.

Question 1.3. Choose any instance with at least 5 cities from the data set for Assignment 1. Choose one strict local optimum found by hill climbing for any problem instance. Confirm that it is a strict local optimum by providing a sample calculation. Recall that a state is a strict local optimum if and only if the best neighbor of the state has strictly higher cost than the current state.

Solution. For instance 7 of a 5-city problem, one solution hill climbing came up with was: [[0, 'A', 63, 90], [4, 'E', 3, 12], [3, 'D', 16, 60], [2, 'C', 20, 99], [1, 'B', 41, 88], [0, 'A', 63, 90]]. This route has a cost of 233.13843662222504. We can confirm this is a local optimum by looking at the neighbour states and confirming they are all higher cost than this state. The neighbour states are given below:

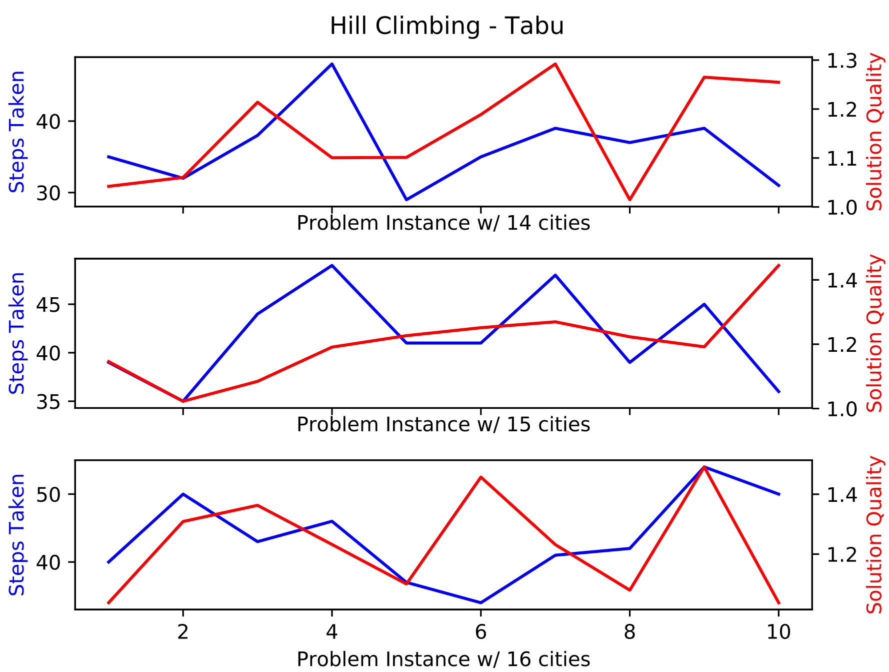
```
[[0, 'A', 63, 90], [3, 'D', 16, 60], [4, 'E', 3, 12], [2, 'C', 20, 99], [1, 'B',
   41, 88], [0, 'A', 63, 90]]
239.9302965397832
[[0, 'A', 63, 90], [2, 'C', 20, 99], [3, 'D', 16, 60], [4, 'E', 3, 12], [1, 'B',
   41, 88], [0, 'A', 63, 90]]
```

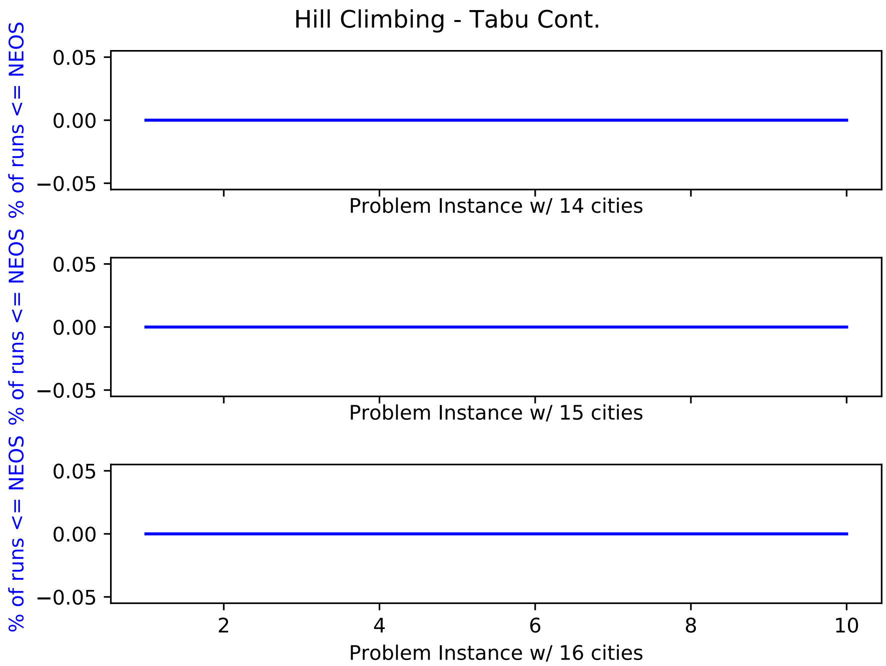
```
239.92692905679488
[[0, 'A', 63, 90], [1, 'B', 41, 88], [3, 'D', 16, 60], [2, 'C', 20, 99], [4, 'E',
    3, 12], [0, 'A', 63, 90]]
285.8846400103083
[[0, 'A', 63, 90], [4, 'E', 3, 12], [2, 'C', 20, 99], [3, 'D', 16, 60], [1, 'B',
   41, 88], [0, 'A', 63, 90]]
285.88464001030843
[[0, 'A', 63, 90], [4, 'E', 3, 12], [1, 'B', 41, 88], [2, 'C', 20, 99], [3, 'D',
   16, 60], [0, 'A', 63, 90]]
302.04743813401967
[[0, 'A', 63, 90], [4, 'E', 3, 12], [3, 'D', 16, 60], [1, 'B', 41, 88], [2, 'C',
   20, 99], [0, 'A', 63, 90]]
253.3115370512497
```

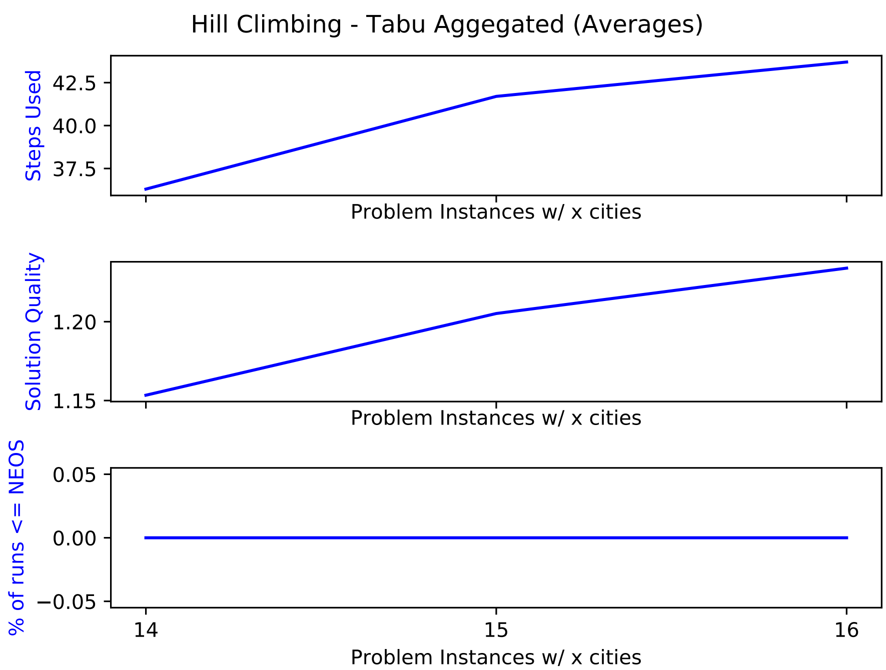
Clearly, since all of these distances are greater than 233.13843662222504, we can conclude the original ordering is, in fact, a local optimum.

Question 1.4. Consider the 14, 15, 16-city instances. Will the hill climbing algorithm perform better if it allows sideways moves and/or maintains a tabu list? Perform an investigation and report your findings. There are many ways to compare the performance of hill climbing with or without sideways moves and/or a tabu list. Feel free to use the examples in part 2 or come up with other performance measures. Be sure to clearly explain what criteria you use to measure the performance of the algorithm. If your answer is yes, provide evidence that adding sideways moves and/or a tabu list improves the performance of hill climbing. If your answer is no, provide evidence that adding sideways moves and/or a tabu list will not improve the performance of hill climbing.

Solution.



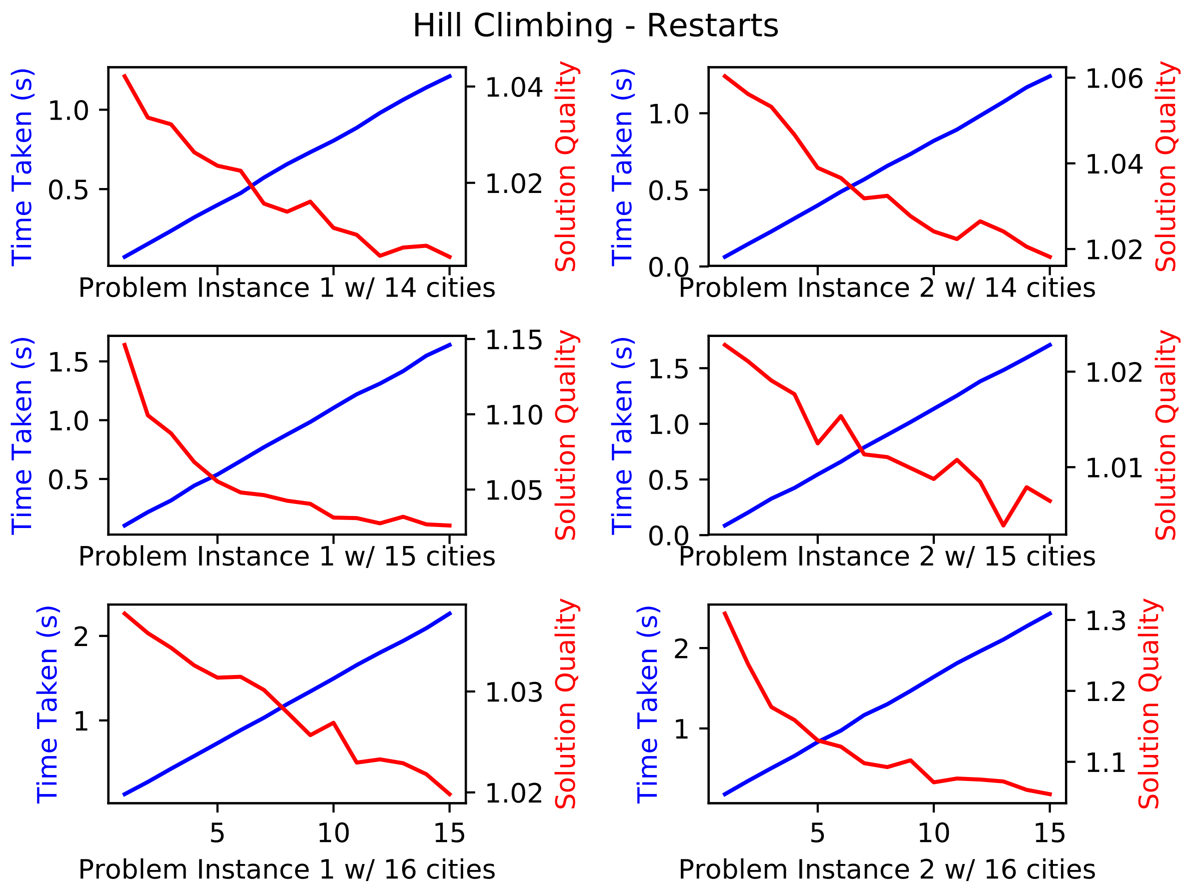




No, introducing tabu lists/sideways moves does not help here, as can be seen in the red solution quality lines (same as question 1.2), and the same-shape solution quality graph in the third set of plots. Sideways moves are almost irrelevant here, since the distances are so precise, and with so many cities, it is very unlikely for a neighbouring ordering to be the exact same length. As for tabu lists, these help avoid getting stuck in plateaus, so considering how unlikely a sideways move is in the first place, this doesn't add much value. This helps explain why the results don't change with this extra logic.

Question 1.5. Consider the 14, 15, 16-city instances. Solve the problem instances using hill climbing with random restarts and report your findings. Consider the first two instances for each number of cities. For each instance, experiment with different numbers of random restarts. For each instance and each number of restarts, run hill climbing with a given number of random restarts for 100 times. For each instance, plot the average solution quality and the average execution time with respect to the number of random restarts (where the average is taken over the 100 repetitions). Then, answer the following questions. For each number of cities, how many restarts is sufficient to ensure that the solution is within 1% of the best solution found by any alternative algorithm? (That is, the quality of the solution is less than or equal to 1.01.) For each number of cities, is there a trade-off between the execution time and the average solution quality? Discuss your observations in no more than one paragraph. Based on your findings, what number of restarts would you choose for each number of cities and why?

Solution.

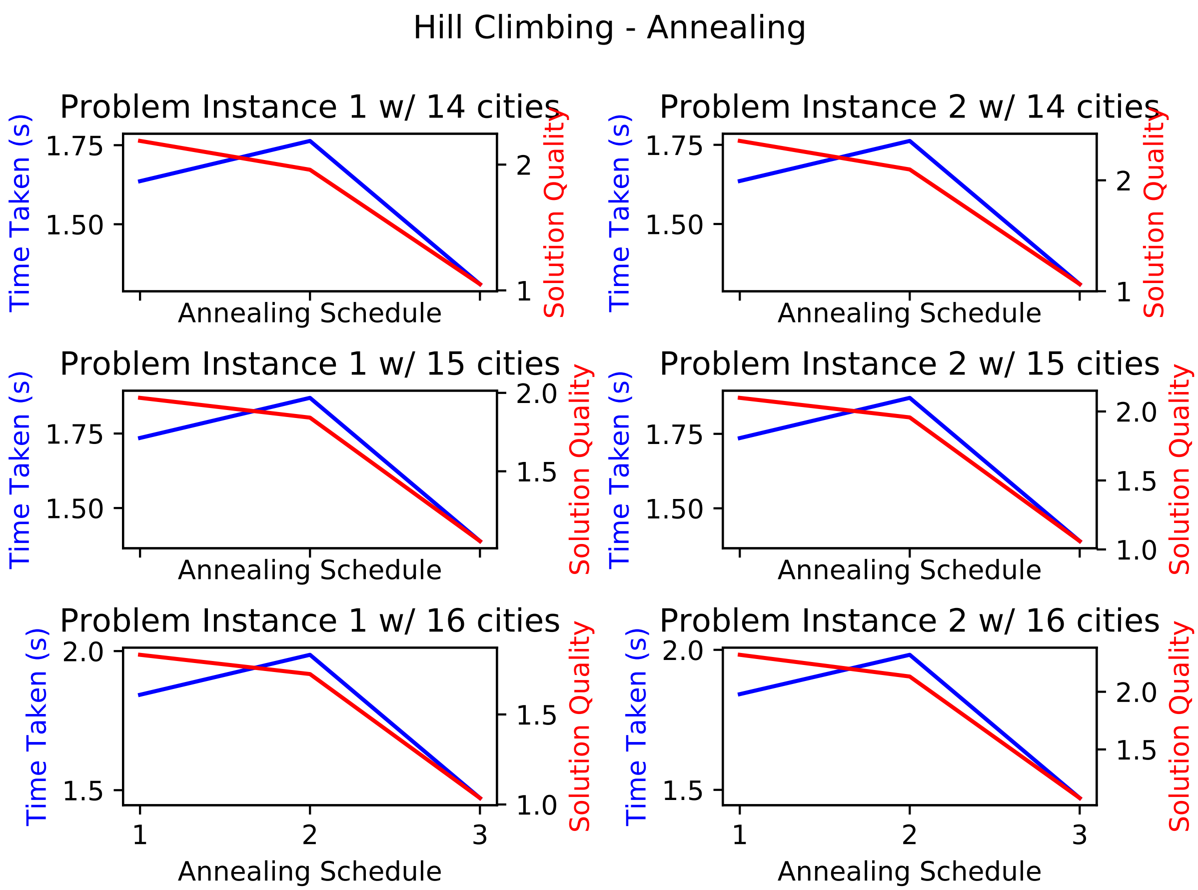


- Unfortunately, I forgot to make proper x-axis labels, but the x-axis represents the number of restarts. I would have re-run it if it didn't take so long with this many restarts...

- For 14 cities, it is 11 restarts. For 15, it depends on the instance, but between 10 and 12 (approximately). For 16 cities, neither the first instance nor the second ever got that close. So, we can conclude ¿15 restarts.
- For each number of cities, there appears to be relatively similar linear trends for both execution time and solution quality. So we will discuss all numbers of cities generally. Execution time is a VERY clear linear increase with more restarts, which is expected. As for solution quality, most trends are of a linear nature, with decreasing difference from the reference (better quality) as restarts increase. A couple show slightly tapering nature as the number of restarts grows. However, we do not have enough data to definitely conclude diminishing returns exist, so the best conclusion we can draw is that as restarts increase, runtime linearly increases, and quality linearly increases as well.
- Based on these results, since each run is still less than 3 seconds long even for 15 restarts, I would say choosing at least 15 restarts is perfectly acceptable, since the runtime is "reasonable", and the quality is better than fewer cities. I cannot definitively recommend more since I don't have data for that, but 15 at the very least is reasonable to my standards, and shows inmprovement over lower quantities.

Question 1.6. Consider the 14, 15, 16-city instances. Solve the problem instances using simulated annealing and report your findings. Consider the first two instances for each number of cities. For each instance, experiment with at least three different annealing schedules. For each instance and each annealing schedule, run simulated annealing for 100 times. For each instance, plot the average solution quality and the average execution time with respect to the annealing schedule (where the average is taken over the 100 repetitions). Then, answer the following questions. For each number of cities, is there a trade-off between the execution time and the choice of annealing schedule? Discuss your observations in no more than one paragraph. For each number of cities, which annealing schedule would you choose for each number of cities and why?

Solution.



- For all cities, the behaviour regarding performance and quality follows the same trend(s). So, we will speak generally here. The quality of the solutions is worst with linear decreases, better with logarithmic decay, and best with exponential decay. On the flipside, while logarithmic decay did offer better solutions than linear decay, it took 5-10% longer, whereas exponential decay not only performed better than both, but took more than 10% less time than linear decay.

- As a result, choosing exponential decay for all numbers of cities would be the logical choice, as it minimizes runtime and maximizes accuracy amongst the available schedules.

Question 1.7. Consider the 36-city instance. Based on your findings above, which local search algorithm would be the best choice to solve the 36-city problem instance? Experiment with different local search algorithms and let each algorithm run for no more than 5 minutes. Report the best algorithm, the best solution found by the algorithm and the cost of the best solution found.

Solution. Based on the above findings, there are 2 potential algorithms that could be selected. For accuracy, random restarts with a high number of restarts seems to be the most accurate. However, this takes very long. The next most accurate algorithm runs much faster, but is not quite as good. This is simulated annealing using exponential decay. Comparing the two using 3 runs of each (all completed in time) to fairly compare the benefits and drawbacks of random restarts to those of annealing, we got a better solution using random restarts (although with much higher runtime). This was with a parameter of 15 restarts per run. The solution cost was 583.7005934569605, and the path was as follows (each element is a city, with the sub-arrays organised as follows [cityID, cityName, X-Coord, Y-Coord]:

```
[[0, 'A', 13, 2], [23, 'X', 46, 35], [21, 'V', 49, 35], [25, 'Z', 51, 18],
[16, 'Q', 40, 14], [9, 'J', 41, 10], [10, 'K', 33, 2], [22, 'W', 33, 5],
[27, 'AB', 23, 25], [13, 'N', 28, 27], [3, 'D', 33, 42], [31, 'AF', 41, 70],
[26, 'AA', 50, 71], [7, 'H', 79, 60], [32, 'AG', 82, 73], [30, 'AE', 99, 81],
[17, 'R', 92, 88], [12, 'M', 80, 91], [34, 'AI', 13, 98], [33, 'AH', 18, 84],
[8, 'I', 27, 76], [24, 'Y', 29, 70], [11, 'L', 26, 67], [19, 'T', 38, 67],
[6, 'G', 61, 57], [1, 'B', 64, 49], [35, 'AJ', 72, 54], [5, 'F', 89, 53],
[18, 'S', 99, 45], [20, 'U', 91, 39], [28, 'AC', 96, 16], [2, 'C', 92, 15],
[4, 'E', 92, 14], [29, 'AD', 92, 7], [14, 'O', 86, 15], [15, 'P', 81, 15],
[0, 'A', 13, 2]]
```

Question 2.1. Calculate the joint probability distribution over the four variables A, B, C, & D. Show the steps of your calculations for at least one probability in the joint distribution

Solution.

- Joint Distribution:
- $P(A \wedge B \wedge C \wedge D) = 0.2016$ This can be shown as: $P(A) * P(B \mid A) * P(C \mid A \wedge B) * P(D \mid A \wedge B \wedge C) = 0.6 * 0.8 * 0.7 * 0.6$
- $P(A \wedge B \wedge C \wedge \neg D) = 0.1344$
- $P(A \wedge B \wedge \neg C \wedge D) = 0.0864$
- $P(A \wedge B \wedge \neg C \wedge \neg D) = 0.0576$
- $P(A \wedge \neg B \wedge C \wedge D) = 0.0216$
- $P(A \wedge \neg B \wedge C \wedge \neg D) = 0.0864$
- $P(A \wedge \neg B \wedge \neg C \wedge D) = 0.0024$
- $P(A \wedge \neg B \wedge \neg C \wedge \neg D) = 0.0086$
- $P(\neg A \wedge B \wedge C \wedge D) = 0.0672$
- $P(\neg A \wedge B \wedge C \wedge \neg D) = 0.0448$
- $P(\neg A \wedge B \wedge \neg C \wedge D) = 0.0288$
- $P(\neg A \wedge B \wedge \neg C \wedge \neg D) = 0.0192$

- $P(\neg A \wedge \neg B \wedge C \wedge D) = 0.0432$
- $P(\neg A \wedge \neg B \wedge C \wedge \neg D) = 0.1728$
- $P(\neg A \wedge \neg B \wedge \neg C \wedge D) = 0.0048$
- $P(\neg A \wedge \neg B \wedge \neg C \wedge \neg D) = 0.0192$

Question 2.2. Calculate the following probabilities. Round your final answers to three decimal places. For each calculation, be sure to show the steps of the calculations using only the probabilities provided or calculated in previous parts.

Solution.

- $P(D)$ = The sum of all the entries in the joint distribution where D is true, which when summed out equals 0.456.
- $P(A \wedge C)$ = The sum of all the entries where A and C are true, which when summed out equals 0.444
- $P(A \mid \neg B) = P(A \wedge \neg B) \div P(\neg B) = 0.119 \div 0.36 = 0.331$
- $P(A \mid B \wedge \neg C) = P(A \wedge B \wedge \neg C) \div P(B \wedge \neg C) = 0.144 \div 0.192 = 0.75$
- $P(A \wedge \neg B \mid C) = P(A \wedge \neg B \wedge C) \div P(C) = 0.108 \div 0.772 = 0.140$
- $P(\neg A \wedge C \mid \neg B \wedge D) = P(\neg A \wedge C \wedge \neg B \wedge D) \div P(\neg B \wedge D) = 0.0432 \div 0.072 = 0.6$

Question 2.3. Based on the probabilities provided and calculated, answer the following questions. Justify your answers by providing probability calculations to prove/disprove unconditional/conditional independence using their definitions.

Solution.

- Are A and C independent?

  Solution. To show independence, we need to show: $P(A \wedge C) = P(A) * P(C)$. First, we calculate $P(A \wedge C)$ by summing from the joint distribution. This give us an answer of 0.444. Then, we need $P(A)$. This is given as 0.6 Now, we need $P(C)$. This is found using the same summation method as before, which gives a result of 0.772. Now, we check the left and right sides of the original equation: $0.444 = 0.6 * 0.772 \rightarrow 0.444 \neq 0.4632$. Since the two sides are NOT equal, C and D are not independent.

- Are A and C conditionally independent given another variable? If so, which variable is it?

Solution.

```
P(A given B) * P(C given B) == P(A and C given B)
P(A given B) = P(A and B) / P(B) = 0.48 / 0.64 = 0.75
P(C given B) = P(C and B) / P(B) = 0.448 / 0.64 = 0.7
P(A and C given B) = P(A and C and B) / P(B) = 0.336 / 0.64 = 0.525
0.75 * 0.7 == 0.525
YAY!
```

As can be seen above, A and C are conditionally independent given B (since $0.75*0.7 = 0.525$).

- Are C and D independent?

Solution. To show independence, we need to show: $P(C \wedge D) = P(C) * P(D)$. First, we calculate $P(C \wedge D)$ by summing from the joint distribution. This give us an answer of 0.3336. Then, we need $P(C)$. This is found using the same summation method, which gives a result of 0.772. So, we need $P(D)$ now, which we got in a previous question as 0.456. Now, we check the left and right sides of the original equation: $0.3336 = 0.772 * 0.456 \rightarrow 0.48 \neq 0.352$. Since the two sides are NOT equal, C and D are not independent.

- Are C and D conditionally independent given another variable? If so, which variable is it?

Solution.

```
P(C given A) * P(D given A) == P(C and D given A)
P(C given A) = P(C and A) / P(A) = 0.444 / 0.6 = 0.74
P(D given A) = P(D and A) / P(A) = 0.312 / 0.6 = 0.52
P(C and D given A) = P(C and D and A) / P(A) = 0.2232 / 0.6 = 0.372
0.74 * 0.52 != 0.372

P(C given B) * P(D given B) == P(C and D given B)
P(C given B) = P(C and B) / P(B) = 0.448 / 0.64 = 0.7
P(D given B) = P(D and B) / P(B) = 0.384 / 0.64 = 0.6
P(C and D given B) = P(C and D and B) / P(B) = 0.2688 / 0.64 = 0.42
0.7 * 0.6 == 0.42
YAY!
```

As can be seen above, C and D are conditionally independent given B (since $0.7*0.6 = 0.42$).

- Are A and B independent?

Solution. To show independence, we need to show: $P(A \wedge B) = P(A) * P(B)$. First, we calculate $P(A \wedge B)$ by summing from the joint distribution. This give us an answer of 0.48. Then, we need $P(A)$. This is given as 0.6. So, we need $P(B) = P(A) * P(B|A) + P(A) * P(B|\neg A) = 0.64$. Now, we check the left and right sides of the original equation: $0.48 = 0.6 * 0.64 \rightarrow 0.48 \neq 0.384$. Since the two sides are NOT equal, A and B are not independent.

- Are A and B conditionally independent given another variable? If so, which variable is it?

Solution.

```
P(A given D) * P(B given D) == P(A and B given D)
P(A given D) = P(A and D) / P(D) = 0.312 / 0.456 = 0.684
P(B given D) = P(B and D) / P(D) = 0.384 / 0.456 = 0.842
P(A and B given D) = P(A and B and D) / P(D) = 0.288 / 0.456 = 0.632
0.684 * 0.842 != 0.632

P(A given C) * P(B given C) == P(A and B given C)
P(A given C) = P(A and C) / P(C) = 0.444 / 0.772 = 0.575
P(B given C) = P(B and C) / P(C) = 0.448 / 0.772 = 0.580
P(A and B given C) = P(A and B and C) / P(C) = 0.336 / 0.772 = 0.435
0.575 * 0.580 != 0.435

P(A given C & D) * P(B given C & D) == P(A and B given C & D)
P(A given C & D) = P(A and C & D) / P(C & D) = 0.2232 / 0.3336 = 0.669
P(B given C & D) = P(B and C & D) / P(C & D) = 0.2688 / 0.3336 = 0.806
P(A and B given C & D) = P(A and B and C & D) / P(C & D) = 0.2016 / 0.3336 = 0.6
0.669 * 0.806 != 0.6
```

As can be concluded by the calculations above, A and B are not conditionally independent.