**INF4490 – Biologically Inspired Computing**

**Assignment 1**

**Name: Furqan Tariq**

**Username: furqant**

**Date: 25-09-2017**

**Instructions**

**The sources codes are written in python language and tested on python 3.6.**

**The organization of source codes as follow:**

**Exhaustive.py has exhaustive solution of problem with simulation (main function)**

**Hillclimbing.py has hill climbing solution with simulation**

**Evolutionary.py has evolutionary solution of TSP problem**

**Util.py contains commonly used functions**

**Pmx.py contains partially crossed over implementation. I used group lecture implementation.**

**Baldwanian.py and Lamarckian.py contains respective implementation of baldwanian and Lamarckian approach.**

**Hybrid.py contains simulation code which simulates both baldwanian and Lamarckian functions.**

**Tests\_output folder contains test outputs data that was generated during my tests**

**Exhaustive Search**

**1.** The shortest tour among the first ten cities is

['Copenhagen', 'Hamburg', 'Brussels', 'Dublin', 'Barcelona', 'Belgrade', 'Istanbul', 'Bucharest', 'Budapest', 'Berlin', 'Copenhagen']

And its distance is 7486.309999999999

**2.** It took 19.1704130173 seconds to calculate it.

**3.** Since, we are checking all the possible combinations that is 24! So it will take very long time

**Hill Climbing**

**1.** The hill climbing algorithm was faster than Exhaustive search. For first ten cities, it took 0.0003903150558 seconds.

For first 10 and 24 cities, the best, worst, mean and deviation of tour length as follow:

|  |  |  |
| --- | --- | --- |
| **No. of Cities** | **10 Cities** | **24 Cities** |
| **Best** | 7503.1 | 21009.17 |
| **Worst** | 11428.9 | 29395.29 |
| **Mean** | 9796.4755 | 25448.856 |
| **Standard Deviation** | 1213.249231 | 2132.276727 |

**Genetic Algorithm**

**1.** In evolutionary approach to solve TSP, I tested it with three different population sizes (that are 50, 200 and 400) for both 10 cities and 24 cities

The results are as follow:

**Parameters i used for this test:**

no\_of\_generation = 1000

no\_of\_parents = 10

mutation\_prob = 0.5

cross\_prob = 1

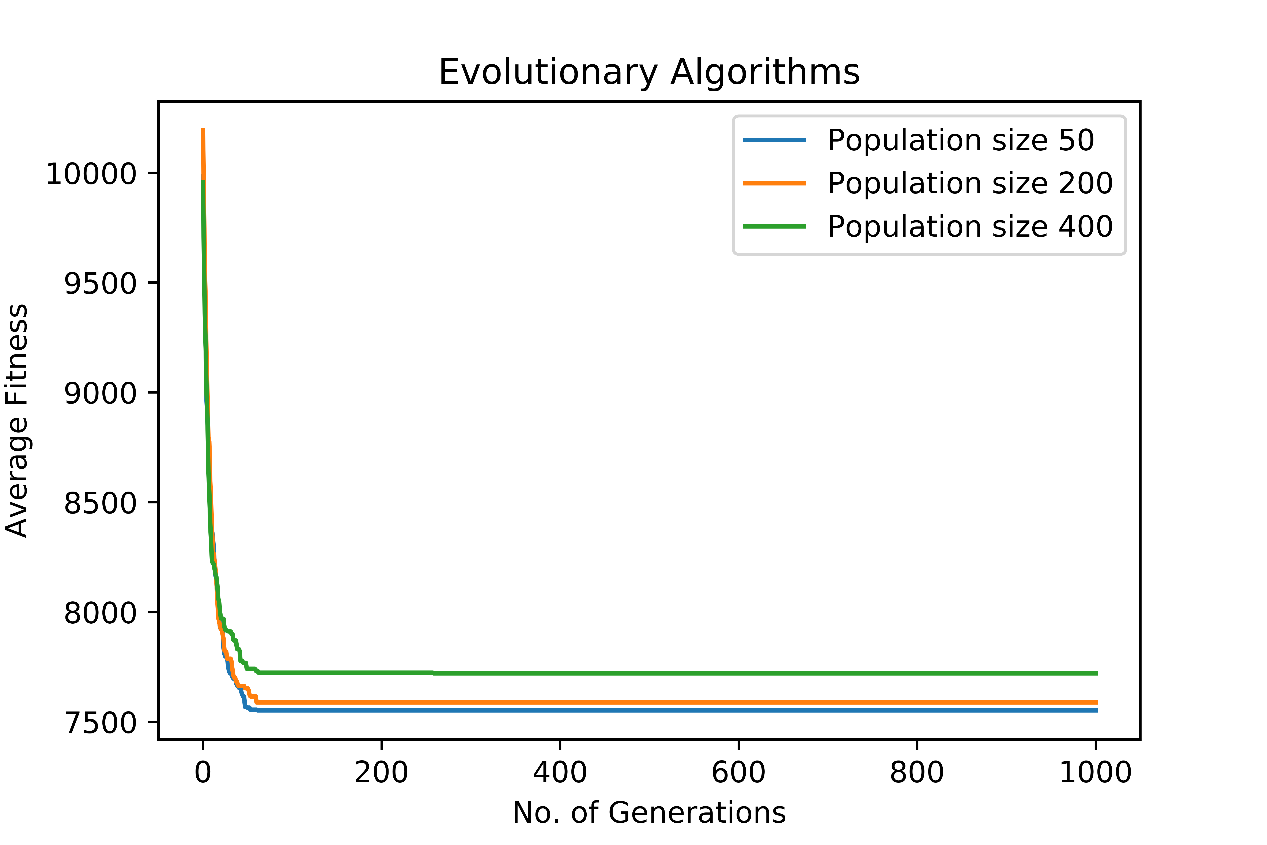
**For 10 Cities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Population Size** | 50 | 200 | 400 |
| **Best** | 7486.31 | 7486.31 | 7486.31 |
| **Worst** | 7737.95 | 8407.18 | 8407.18 |
| **Mean** | 7551.7385 | 7593.055789 | 7720.3695 |
| **Standard Deviation** | 110.4634705 | 221.9215492 | 356.3565585 |

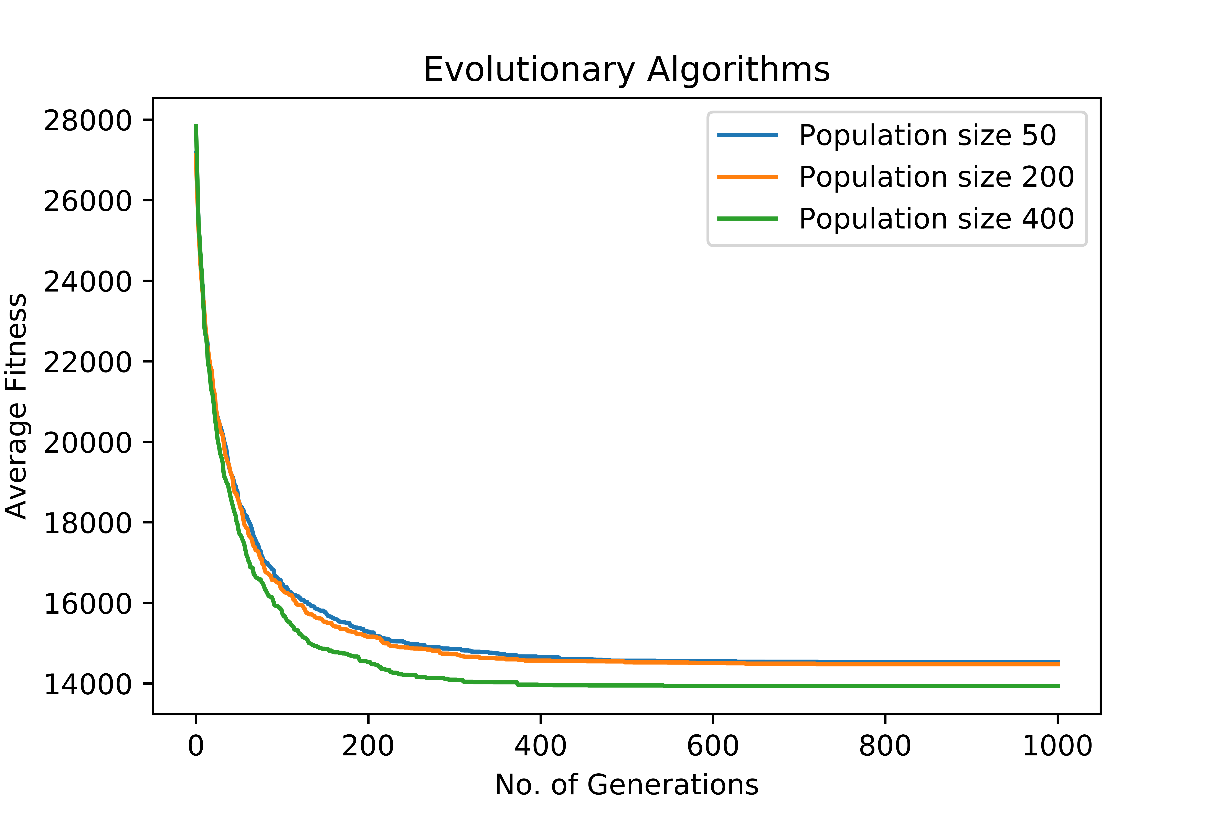
**For 24 Cities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Population size** | 50 | 200 | 400 |
| **Best** | 12561.22 | 12853.95 | 12416.87 |
| **Worst** | 16274.58 | 16223.61 | 15685.79 |
| **Mean** | 14526.1075 | 14439.42579 | 13852.15368 |
| **Standard Deviation** | 1005.72295 | 818.2993311 | 935.9036519 |

**Generational Plots For 10 Cities**



**Generational Plot For 24 Cities**

****

**Conclusion From Data**

In case of 10 cities, 50 population is more suitable because it found a correct result and also it took lower number of evaluation. But in case of 24 cities, The 400 population is more suitable because it came close to the actual solution and the generation graph is showing that in every generation it is producing optimal results.

I am using distance as a fitness function so lower distance represents fittest solution.

**2.** Yes it did find the correct shortest distance among first 10 cities as found by exhaustive search which is 7486.31.

**3.** Yes it was significantly faster than exhaustive search.

The table of average time taken in execution is as follow:

|  |  |  |
| --- | --- | --- |
|  | **10 Cities** | **24 Cities** |
| **Population size 50** | 0.1375925899 seconds | 0.2348968744 seconds |
| **Population size 200** | 0.1556320818 seconds | 0.2571892989 seconds |
| **Population size 400** | 0.1760002971 seconds | 0.2813682054 seconds |

**4.** In exhaustive search, the number of tours examined were 24!

But in evolutionary algorithm, the number of tours examined should be equal to the number of generations multiply by population size in each generation.

In my case, it is 400\*1000 which is far less than 24!.

**Hybrid Algorithm**

By using hybrid approach the best, worst, mean and deviation of tour length as follow:

**Parameters i used for this test:**

no\_of\_generation = 10

no\_of\_parents = 10

mutation\_prob = 0.5

cross\_prob = 1

population\_size = 200

hill\_climb\_iteration = 5

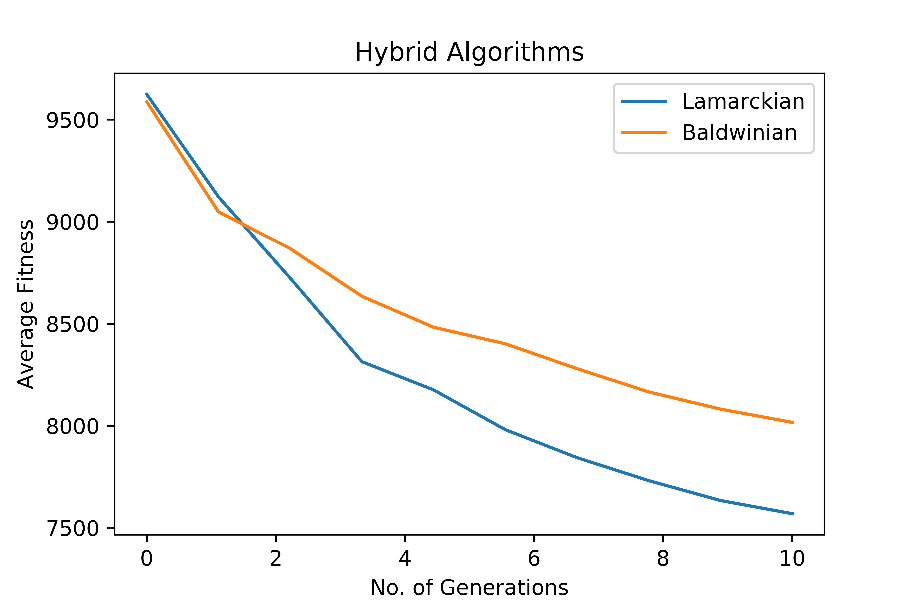
**For 10 Cities**

|  |  |  |
| --- | --- | --- |
| **Learning Models** | **Baldwinian** | **Lamarckian** |
| **Best** | 7663.7 | 7486.31 |
| **Worst** | 10799.72 | 8204.06 |
| **Mean** | 8847.08 | 7569.279 |
| **Standard Deviation** | 987.928563 | 174.7544586 |

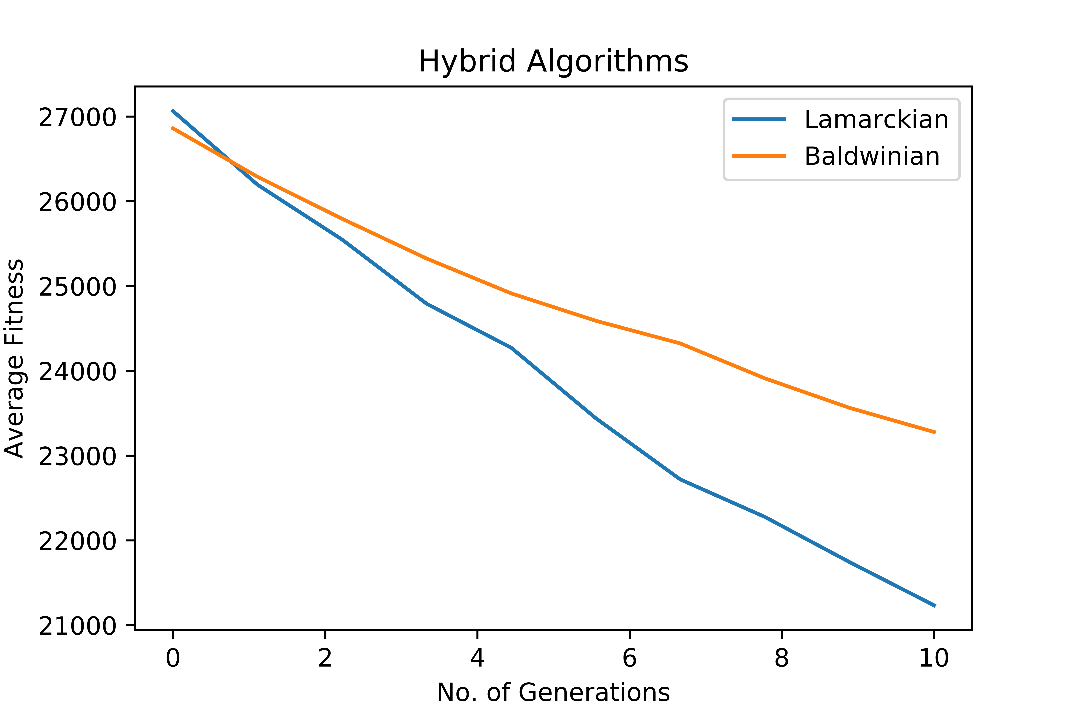
**For 24 Cities**

|  |  |  |
| --- | --- | --- |
|  | **Baldwinian** | **Lamarckian** |
| **Best** | 21655.07 | 18367.97 |
| **Worst** | 27731.36 | 22691.91 |
| **Mean** | 24594.07222 | 21234.1605 |
| **Standard Deviation** | 1448.110643 | 1247.608216 |

**Generational Plot for 10 Cities**

****

**Generational Plot for 24 Cities**

****

**1.** In hybrid solution, the number of evaluation is equal to the production of no\_of\_generations, population\_size and hil\_climb\_iteration. In my case it is 10\*200\*5 which is less than pure GA approach and found a correct solution.

I couldn’t implement programmatically way to calculate number of evaluation due to the misunderstanding of it and shortage of time.