FINAL PROJECT

CMPT310 Surrey, Spring 2020

Phan Bui

301325875

dbui@sfu.ca

❖ Framework

I use totally 4 frameworks in this project:

- PMX and mutation: I got this idea from note from course website at https://www2.cs.sfu.ca/CourseCentral/310/tjd/geneticAlgorithmNotes.html. To run this algorithm, open terminal in the folder and command python3 project.py.
- Reproduce and mutation: I think of this idea by myself. To run this function, modify run() function in project.py by commenting everything and uncomment 3 lines from 457 to 459. Open terminal in the folder and command python3 project.py.
- PMX1 and mutation: I got this idea from a website at https://www.hindawi.com/journals/cin/2017/7430125/. To run this function, modify run() function in project.py by commenting everything and uncomment 3 lines from 460 to 462. Open terminal in the folder and command python3 project.py.
- Mutate Search: I got this idea from note from course website at https://www2.cs.sfu.ca/CourseCentral/310/tjd/geneticAlgorithmNotes.html. To run this function, modify run() function in project.py by commenting everything and uncomment 3 lines from 463 to 465. Open terminal in the folder and command python3 project.py.

❖ Ideas

I experiment with 4 algorithms:

- PMX and mutation: This algorithm is explained in the course note. Assume that population size is n. For n/2 time, randomly select 2 parents, parents with shorter travel distance has more chance to be chosen. 2 parents will make 2 children using PMX. Mutate the 2 children with the probability decided by the user and add it to the next generation.
- Reproduce and mutation: I feel that PMX and does not represent normal behavior, it implies that every time parents give birth to 2 children. Hence, I come up with function reproduce. Assume that population size is n. For n time, randomly select 2 parents, parents with shorter travel distance has more chance to be chosen. 2 parents will give birth to 1 child only. The algorithm will select a random fragment from one parent, copy it to the child. The algorithm then scans the other parent, if the city does not exist in the child, copy the city into the child. From a current generation, if a route has the best travel distance, copy the route into the next generation. If a route does not have best travel distance, mutate the route with the probability decided by the user and add it to the next generation.

- PMX1 and mutation: Assume that population size is n. For n/2 time, randomly select 2 parents, parents with shorter travel distance has more chance to be chosen. 2 parents will make 2 children. The idea is provided in section 2.1 in https://www.hindawi.com/journals/cin/2017/7430125/. Mutate the 2 children with the probability decided by the user and add it to the next generation.
- Mutate Search: This algorithm is explained in the course note. Assume that population size is n. For n time, mutate a deep copy of the best route in the generation using provided probability and add it to the next generation.

Discussing runtime: for each algorithm, I experiment 2 population size 20 and 100.

- I see that algorithms with population 20 run faster than with population 100. This is no surprise since the processor needs more time to process. PMX and mutation runs about 2 times slower. Reproduce and mutate runs about more than 3 times slower. PMX1 and mutation runs about 3 times slower. Mutate search runs about 3 times slower.
- I see that reproduce and mutation run the slowest. This is because 2 parents only produce 1 child at a time. PMX and mutation and PMX1 and mutation are almost similar in idea. However, PMX1 and mutation runs a little slower because recursion is employed. Mutate search runs the fastest. This is because it is the simplest one, 1 parent make the entire population in 1 iteration, implementing the algorithm require least loop among all algorithms.

Discussing performance:

- I see that when I start running algorithms, small population improve the result slower than large population. However, large population stop making significant improvement sooner than small population. I think this is because in the first place, with large population, algorithms make more calculation, the chance of improving is hence increased. This explain why large population is more aggressive in the beginning. However, in the long run, large population has a trade-off, it takes more time to compute. In the same amount time, it is unsure small or large population is better. However, in same amount of iteration (generation), I believe that large population will give better result than small population.
- Among all algorithm. PMX and mutation, reproduce and mutation, PMX1 and mutation perform about the same, they cannot improve at around travel distance of 1 million. Mutation search perform the best, giving travel distance of about 300 thousand. This really surprises me. I haven't yet come up with a suitable explanation for this phenomenon. All I can think of is that there is much more randomness in mutation search. The algorithm is much less predictable

than others, it will bring more extreme values. The algorithm can give very long travel distance, or very small travel distance just by luck.

❖ Challenge Problem Results

- I obtain best result by using mutate search algorithm. Assume that population size is n. For n time, mutate a deep copy of the best route in the generation using provided probability and add it to the next generation. I use population 20, mutate probability 0.8, runtime is more than 1 hour, the shortest route I found is 314753.82030235743.

❖ Notes

- To run test for each function I write, uncomment line 467, comment line 468.
- I save my result in files. 20mutateSearchLog.txt is what in my terminal when I run mutate search for population 20. 20mutateSearchResult.txt is the result route when I run mutate search for population 20.
- In function run(), user can change file, populationSize, mutateProbability, estimatedAllowedTime. These are parameters for algorithms.

Reference:

https://www2.cs.sfu.ca/CourseCentral/310/tjd/geneticAlgorithmNotes.html

https://www.hindawi.com/journals/cin/2017/7430125/