Simulated Annealing and Genetic Algorithm implementations for Symetric Travelling Salesman Problem (ATSP)

Encut Octavian Ploae Radu, grupa 2A1

January 14, 2022

Abstract

This report contains the results of two metaheuristic algorithms solving The Traveling Salesman optimization problem. Two diffent algorithms will be compared: a probabilistic one (Simulated Annealing) and an evolutionary one (Genetic Algorithm). The Genetic Algorithm should get better results most of the time.

1 Introduction

In this paper we will observe the results of non-deterministic algorithms when trying to find the optimum path for different instances of the Symmetric variant of the Traveling Salesman Problem. We are only going to be using paths that have the edge weight type in the Euclidian 2D space. For this we are going to be using two different algorithms and comparing them to see which one is better for every instance, these algorithms being Simulated-Annealing and a variant of a Genetic Algorithm.

2 Methods

We are going to be using two algorithms, the Simulated-Annealing algorithm and a Genetic Algorithm.

2.1 Simulated-Annealing

Simulated annealing is a probabilistic technique for approximating the global optimum of a given function.

The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to alter its molecular structure and improve it. The idea behind this implementation is that a high temperature allows jumps out of a local basins trading a better solution now for the chance to get a better one later in the "annealing" process.

The algorithm starts by arbitrarily selecting a candidate solution and an initial temperature. Then we must have two conditions: the termination condition(the number of steps per temperature) and halting criterion(the steps it takes for the temperature to reach equilibrium). Inside the inner loop a random hamming neighbour of the arbitrarily selected solution is selected as a canditate and evaluated. If it is better, it becomes the new solution and if it's not better we select a random number from 0 to 1 and using a forumula we calculate the probability of the process to go uphill. If the random number we selected is smaller, the solution becomes the candidate even if it's a worse fit for now. The probability of accepting a worse solution decrease with the temperature.

2.2 Genetic Algorithm

In computer science and operations research, a genetic algorithm (GA) is a metaheuristic inspired by the process of natural selection. Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover and selection. In a genetic algorithm, a population of candidate solutions (called

individuals) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The algorithm starts by arbitrarily selecting a population , usually the population has a size of several hundred individuals. After the initial selection, a rank is given to each individual, this rank is called "fitness" and it represents how fit an individual is to live to the next generation or create offspring. After that a selection is made in which a couple individuals are chosen, either completely randomly or based on how "fit" they are. Those individuals will create offspring using a system called crossover in which genes from each parent is selected, but like in real life mistakes can happen, and those mistakes are represented my mutation. Each children has a low probability to mutate, meaning that one of the chromosomes will change to create a more diverse population. Those children will replace the current population, creating a new generation. Then we give a fitness rank to each individual and the cycle continues until a termination criteria is reached.

2.3 Implementation details

Software and hardware used:

- All algorithms were written in Python
- All algorithms were ran using either a personal computer or Google Colaboratory.

Simulated-Annealing

- The solution is represented by coordinates ordered such the the path taken is the smallest it can be
- The initial temperature is set to 1000
- Temperature decreases by 1%
- The halting criterion is temperature <10e-5
- The termination condition is i <1000
- The precision chosen for representing the solution and candidate solution is 10e-5

Genetic Algorithm

- The population is made out of 2000 individuals
- The fitness is created by taking the multiplicative inverse of the solution
- The selection is made by choosing the best 200 individuals in a generation based on fitness
- The crossover is done by OX crossover
- The probability to mutate is 1%
- The termination condition is reaching 1000 generations
- The precision chosen for representing the solution and candidate solution is 10e-5

Size	Expected value	Mean	Min	Max	Avg Time(s)	St_{dev}
berlin52	7542	8032.74687	8003.35793	8061.74683	1861.25831	19.78616
ch150	6528	7772.14217	7763.21251	7780.96548	3017.8390	5.9975
pr107	44303	56826.1165	55562.6033	58213.0247	2456.2602	798.367
bier127	118282	135181.190	134618.814	135610.489	2888.96672	306.5371
lin105	14379	19076.2118	18910.5953	19230.2257	2650.6205	103.9467
pr439	107217	192469.5497	161595.0547	209070.8861	6536.8932	16666.5501
pr264	49135	75971.2066	70565.6759	79337.0308	3536.6173	2590.1595
linhp318	41345	67248.1827	65952.4050	68304.6211	5328.4712	767.5481
rat195	2323	7080.2279	6835.505	7286.6539	3622.1641	151.3349
kroA200	29368	39545.1479	37452.9651	42243.0504	3865.8431	1536.1031

Table 1: Simulated-Annealing

Size	Expected value	Mean	Min	Max	Avg Time(s)	St_{dev}
berlin52	7542	7883.5123	7554.04511	8126.39425	539.8059	161.967873
ch150	6528	8674.90183	6529.41420	10898.63821	1765.1651	1497.17093
pr107	44303	48104.3272	44431.00471	52891.75181	1253.28450	2608.73269
bier127	118282	123268.728	118611.91416	129873.97546	1432.5136	3155.83905
lin105	14379	16724.2442	14614.29715	18968.54369	1245.5823	1448.91782
pr439	107217	160722.604	114257.81196	196441.5798	5435.2732	25067.9875
pr264	49135	60038.0579	52309.4653	69727.6019	2543.3867	6024.3815
linhp318	41345	52518.9391	42539.7332	59780.2407	4623.3287	4926.9902
rat195	2323	3740.44899	2418.6947	4949.6066	2104.2334	794.7240
kroA200	29368	38779.9557	29626.77364	49783.92192	2324.5935	6835.166

Table 2: Genetic Algorithm

Observations

Looking at these tables we can deduce that the Genetic Algorithm runs faster than the simulated annealing while generating decent results, but because the implementation we have for the genetic algorithm is really greedy when the instances get difficult the results get worse because the algorithm isn't exploring as much as is should. For smaller instances the results we get are pretty good, and in most cases it beats the simulated annealing in time and results even if they are pretty close to each other.

3 Conclusion

This paper compared Simulated Annealing and Genetic Algorithms in a multitude of situations and showed both the advantages and disadvantages of implementing a non-deterministic algorithm. On one hand these algorithms are relatively easy to implement, are faster is most situations and can produce accurate results. On the other hand the results they generate aren't always accurate, even if they do get pretty close to the actual result. The genetic algorithm is better than the simulated annealing in this case because it runs in a very short amount of time and generates decent results in most situations, even if the results usually vary a bit from run to run. Better results are possible, especially for the genetic algorithm because there are a lot of things that we can change, but this paper proved that, even without a lot of tuning, in most situations this type of algorithm is a better fit than simulated annealing for this optimisation problem.

Biblography

1. Jon Fincher - Reading and Writing CSV Files in Python https://realpython.com/python-csv
2. Wikipedia - Simulated annealing https://en.wikipedia.org/wiki/Simulated_annealing
3. Eugen Croitoru - Genetic Algorithms https://profs.info.uaic.ro/~eugennc/teaching/ga/
4. https://www.overleaf.com
5. Dr. Trefor Bazett - Intro to LaTeX : Learn to write beautiful math equations https://www.youtube.com/watch?v=Jp0lPj2-DQA
6. Neal Holtschulte,Melanie Moses - Should Every Man be an Island? https://www.cs.unm.edu/~neal.holts/dga/paper/holtschulte2013island.pdf
7. https://latex-tutorial.com/tutorials/
8. Towards Data Science - Introduction to Genetic Algorithms Including Example Code https://bit.ly/32bGIRU
9. Wikipedia - Genetic Algorithms https://en.wikipedia.org/wiki/Genetic_algorithm
10. GeeksforGeeks-Genetic Algorithms https://www.geeksforgeeks.org/genetic-algorithms/
11. Jason Brownlee - Simple Genetic Algorithm From Scratch in Python https://machinelearningmastery.com/simple-genetic-algorithm-from-scratch-in-python/
12. PyGAD - Python Genetic Algorithm https://pygad.readthedocs.io/en/latest/
13. Towards Data Science - Evolution of a salesman: A complete genetic algorithm tutorial for Python https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35
14. Wikipedia - TSP https://en.wikipedia.org/wiki/Travelling_salesman_problem
15. Dhawal Thakkar - Using a Genetic Algorithm for Traveling Salesman Problem in Python https://crescointl.com/2021/03/17/using-a-genetic-algorithm-for-traveling-salesman-problem-in-python/