UNIVERSITÉ
DE GENÈVE

GENEVA SCHOOL OF ECONOMICS
AND MANAGEMENT

The TSP with a Genetic Algorithm

A basic and improved algorithm to avoid local optima

Semester project for the Numerical Optimization and Simulation course (S403110)

Kieran Schubert
07.01.2019

Professor: Dietmar Maringer
Teaching Assistant: Katarzyna Reluga

Introduction

A few weeks ago, newspapers revealed that officers of the swiss military high command were caught flying their wives by helicopter across the country to attend gala dinners. Combined with knowledge of the Travelling Salesman Problem, this prompted me to think about how they could have optimized their route if they had decided, say for New Year's Eve celebrations, to attend a gala in each swiss cantonal capital in one night. What would have been the shortest route?

Setup

The setup of the problem is the following: what is the shortest path between the 26 swiss cantonal capitals starting from and rejoining the same city in the end. By obtaining GPS locations of each city, the node positions were calculated by approximating their coordinates for a plane. All routes between nodes are straight lines computed as the Euclidian distance in kilometers between the two points.

Algorithm

Different methods exist to tackle the TSP, such as exact, heuristic or approximation algorithms. Exact solutions such as the use of brute force would run with time O(n!), which is already unfeasible on a personal laptop for 26 nodes. Heuristic methods usually yield slightly sub-optimal solutions but can run in polynomial time. This class contains among others the Ant Colony Optimization, Genetic Algorithm, k-opt, Nearest Neighbour and Simulated Annealing methods. The last class of approximation methods containing the Christofides algorithm will not be discussed here. This exercise will focus on the Genetic Algorithm.

The Genetic Algorithm is inspired by nature, namely selecting parent solutions to generate offspring by transfer of genotype (DNA) using crossover and mutation and choosing the best candidate solutions from a selection pool by some selection criterion. Solutions are the expressed genotype, or phenotype.

GA pseudocode

Generate initial parent P1
Evaluate fitness of P1
For *max number of generations* ***do***
    Generate parent P2
    Generate offspring by crossover between P1-P2 and mutation
    Evaluate fitness of offspring
    Select best of offspring vs parent who becomes parent P1
End

A basic version of the Genetic Algorithm studied in class was modified to accommodate the problem. Indeed, the problem arises that each city must be visited only once with the exception of the starting/ending city. Classical crossover doesn't account for the unicity of transmitted DNA, resulting in children inheriting the same city multiple times. Classical mutation also causes the same cities to sometimes appear more than once in the offspring.

Ordered crossover and Swap mutation were employed to tackle this issue. Many other variants for crossover and mutation exist[1] but will not be discussed here.

Features and parameters

Ordered Crossover: a subset of parent 1 is selected from a random cutoff point. One child is generated by inheriting this subset (with the same order) at the same cutoff position. The transmitted subset is compared to the elements of parent 2 in order to select the elements that are not common. These elements are then transmitted to the child.

Swap Mutation: The order of two cities are swapped at random for the children which are selected for mutation.

Fitness measure: the fitness function evaluates the Euclidian distance for the whole route. The individuals with the lowest distance are deemed the fittest and are thus more likely to be selected for reproduction.

Population size: The population size determines the number of individuals per generation. A large population size may prevent premature convergence to local optima.

Selection: Selection between offspring and parents is based on the tournament method, thus giving the possibility to individuals with lower fitness to be selected in order to preserve some genetic diversity.

Local optima

A standard Genetic Algorithm may sometimes converge prematurely to local optima. This problem arises when the genetic diversity of both parents is insufficient to generate better solutions than the local optima. Different methods are discussed in the following section to avoid this issue.

AMR: Adaptive Mutation Rate[2]:A seemingly standard method would be to increase the mutation rate with generations in order to "jiggle" out of local optima by randomness once the genetic diversity of parents becomes too low. An implementation relying on this technique is similar to a reversed cooling method as seen is *Simulated Annealing*. The main advantage would be a self-calibrating mutation rate, which the algorithm user would not have to worry about calibrating.

ROG: Random Offspring Generation[3]*:* When a situation with a sufficient loss of diversity in parents occurs, the probability of offspring being genetically similar to the parents is increased. At least two methods can be imagined here: firstly, if both parents have similar phenotypes, the offspring is generated at random (*ROG 1)*, or secondly, the offspring inherits a subset of parent one and the crossover occurs with a randomized subset of parent 2 (*ROG 2)*.

[1] Potvin, J.-Y., Genetic Algorithms for the traveling salesman problem

[2] Rocha, M., Neves, J., Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation

[3] Ibid.

FVTLI: Four Vertices and Three Lines Inequality[4]*:* After evaluating the fitness of children, subsets of four sequential cities are extracted from all offspring and the central cities (position 2 and 3) are flipped. Fitness is then evaluated again, and the improved offspring are selected over their counterpart *(FVTLI 1)*. This method only requires the computation of the distance between two new points. A more general approach consists in selecting sequential groups from 4 to N-3 elements (N is the number of nodes) and then flipping the order of cities. Fitness is then evaluated and compared to the original offspring *(FVTLI 2).* The selected modifications are the *AMR, ROG 2* and *FVTLI 2*. The *ROG 1* is not used as it is possible it will introduce too much noise and will not help convergence. The *FVTLI 2* is preferred over *FVTLI 1* as it is a more general technique.

Optimized Genetic Algorithm pseudocode

Generate initial parent P1
Evaluate fitness of parent
For *max number of generations* ***do***
    Increase mutation rate with gen *(AMR)*
    Generate parent P2
    Check if parents are genetically different *(ROG 2)*
    If not **randomize** parent P2
    Generate offspring by crossover and mutation
    Select consecutive subsets of offspring and flip order *(FVTLI 2)*
    Evaluate fitness of offspring
    Select best of offspring vs parent who becomes parent P1
End

Analysis of results

Crossover rates of 0.5 and mutation rates of 0.01 were used with 100, 200, 300 and 500 maximum generations. The population size is set to 50. Results are showed in *Figure 1* and *Table 1* for the basic Genetic Algorithm with 20 experiments per maximum generation length.

MaxGen	100	200	300	500
Min. distance [km]	986.4535	955.5730	967.2474	948.7177

Table 1.

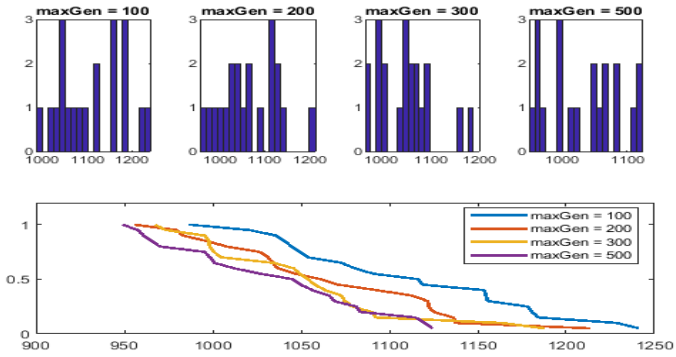

Figure 1. Basic GA results

[4] Borna, K., Hashemi, V. H., An improved genetic algorithm with a local optimization strategy and an extra mutation level for solving Traveling Salesman Problem

After many trials, it would seem the global minimum is at 940.4061 km. Although the basic algorithm doesn't find the global optimum rigorously given a low number of maximum generations, it does get close given a sufficient amount of generations. Results with 100 to 200 iterations are nevertheless within 5% of the optimal solution, giving the user a rough idea of the global optimum. A quick look at the parents shows that genetic diversity is quickly lost and the algorithm gets stuck in a local optimum. Based on these results, the Genetic Algorithm with features to prevent premature convergence to local optima was implemented using the same parameters, and its results are presented in *Figure 2* and *Table 2*.

MaxGen	100	200	300	500
Min. distance [km]	951.2242	940.8877	950.7973	940.4061

Table 2.

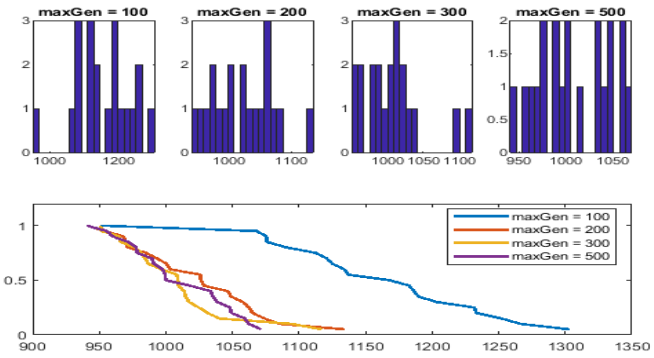

Figure 2. Optimized GA results

Results with the modified Genetic Algorithm are slightly better than with the basic version, showing that indeed the extra features are useful. This version converges to the global optimum in less than 500 iterations but solutions are still spread out. In an effort to obtain better results, the basic Genetic Algorithm parameters were fine-tuned. The results are presented in *Figure 3* and *Table 3* for crossover rates of 0.5, mutation rates of 0.001 and a population size of 1000.

	Without *ROG*	With *ROG*
MaxGen	200	200
Min. distance [km]	940.4061	940.4061

Table 3.

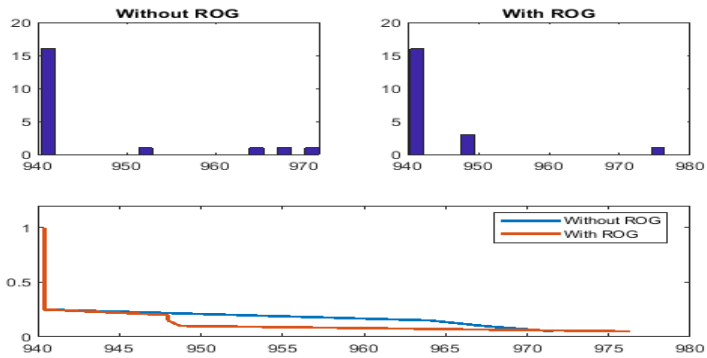

Figure 3. Basic GA with tuned parameters

It is clear that the parameter selection is key to the use of the Genetic Algorithm. This version rigorously converges to the global optimum in less than 200 generations, with a slightly better performance for the version with *ROG*. These results are the most promising as they rely on a simpler algorithm although the computing power required to evaluate a high number of individuals is increased.

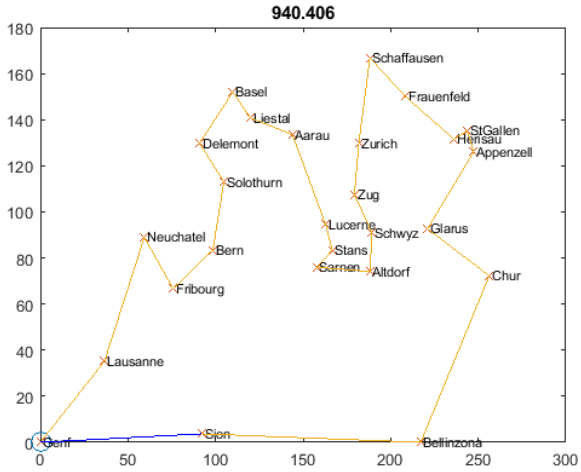

Figure 4. Optimal route

Conclusion

This work has showed that the Genetic Algorithm is a suitable method to solve shortest path problems. However, the issue of premature convergence to local optima due to loss of genetic diversity has to be addressed. A basic version is sufficient to obtain an idea of the global optimum, whereas one might consider extra features to increase solution quality. Many extensions exist and can be implemented to avoid this problem, such as *Adaptive Mutation Rates*, *Random Offspring Generation* or *Four Vertices and Three Lines Inequality*. However, much better convergence is observed by simply tuning the algorithm parameters, most importantly the population size bearing in mind the extra computational cost required. This obviously opens the door to parameter optimization and *meta-heuristic* or even *hyper-heuristic* methods. An individual willing to tackle larger scale shortest path problems may want to consider a combination of the methods presented in this exercise.

References

Borna, K., Hashemi, V. H., An improved genetic algorithm with a local optimization strategy and an extra mutation level for solving Traveling Salesman Problem, *International Journal of Computer Science*, Engineering and Information Technology (IJCSEIT), Vol. 4, No.4, August 2014

Rocha, M., Neves, J., Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation, *International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*, 1999, Multiple Approaches to Intelligent Systems, pp. 127-136

Potvin, J.-Y., Genetic Algorithms for the traveling salesman problem, *Annals of Operations Research*, Vol. 63, 1996, pp. 339-370