A Genetic Algorithm to Autonomous Vehicles Designation

Erick de Barros Alcântara, Marconi de Arruda Pereira

[1]Departamento de Tecnologia e Eng. Civil, Computação e Humanidades
Universidade Federal de São João Del Rei - Campus Alto Paraopeba
MG 443, KM 7 Ouro Branco – MG – Brazil

eryckbarros@hotmail.com, marconi@ufsj.edu.br

Abstract. *This paper proposes a Genetic Algorithm that can be used as an alternative designation method for autonomous vehicles. The proposed method is compared with some popular algorithms used in vehicle designation, like the Hungarian Algorithm and a Greedy Algorithm. The main goal of this paper is to obtain a method with faster execution time of that obtained by the Hungarian Algorithm and a better performance than the considered Greedy Algorithm. The proposed method was tested in a simulation scenario used in other works. The results indicates that the proposed algorithm is capable of generating promising results, since the execution time was reduced by 70% compared to the Hungarian Algorithm, also presenting a better response than the Greedy Algorithm.*

1. Introduction

This paper presents a study about vehicle designation, as used in taxi and Uber services, that can be used by autonomous vehicles in the future. The users can demand a vehicle in any point of the city, at any time. On the other hand, the vehicles are moving across the streets, attending or waiting for a new call. It the past, in the context of taxis, the user called the vehicle center and requested the service; then, the attendant made a radio call for all vehicles and the driver who considered himself closest to the customer answered the call. Later, with the popularization of smartphones, several applications were released to identify the position of customers and vehicles, in order to allocate the nearest vehicle to perform the service. In this last context, there are many possible strategies. A Greedy Algorithm (GA) is a common method to perform this designation. It consists of designate the the closest vehicle to the demand point, considering the Euclidean distance between them. A improvement can be done, replacing the Euclidean distance with the actual distance (i.e., the distance that GPS equipment indicates between the user and the car) that the vehicle needs to travel to find the customer [Reis et al. 2013].

Other works [Oliveira et al. 2015, Souza et al. 2016] point out that an optimization model can be applied, based on the designation problem [Cormen et al. 2009]. The contribution then consists not only in finding the best vehicle to answer a particular call, but in allocating the available vehicles to the existing demand, minimizing the total distance that the cars must travel. The Hungarian Algorithm (HA) gives the optimal value [Kuhn 1955, Jonker and Volgenant 1986], but, in most cases, takes a long execution time.

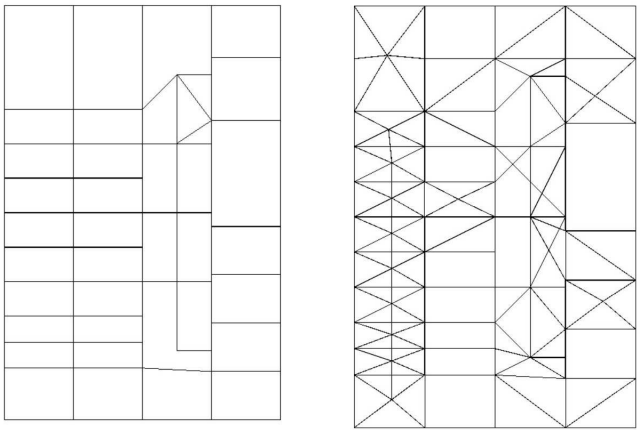
The main contribution of this paper is the proposal of a vehicle designation algorithm, which takes into consideration real traffic conditions, that presents better answers than a greedy approach and that generates results in an acceptable time. It is proposed

a Genetic Algorithm (GE) [Goldberg and Holland 1988] based tool, with genetics' operators specifics to vehicle designation problem. The proposed method were tested in a traffic simulator tool that were configured to reproduce the environment that vehicles can face to meet the users. The results indicates that the proposed algorithm can obtain better results than the GA, generating the results in a feasible computing time, specially in a complex scenario, as are maps of medium and large cities.

2. The Simulation Environment

SUMO[1] is an open source traffic simulation package [Behrisch et al. 2011], that offers a huge variety of tools to simulate traffic elements like pedestrians, conventional cars, buses, autonomous car, electric car and even traffic lights. In this paper, SUMO was used to simulate a city map, which consists in a modification of a previous version (Figure 1(a)), presented in [Oliveira et al. 2015]. In Figure 1(b) it is presented the used map.

The simulation environment were configured using the following parameters: 850 autonomous vehicles (AV) to answer the simulated calls, and 18,000 other vehicles to simulate the traffic flow, set in 50,000 different routes. All AVs were added to the simulation right on the beginning, to ensure that calls made early could be answered. The remaining vehicles (private cars and buses) were added gradually as the simulation runs, preventing an excessive traffic jam on the entrance points of the simulation.



(a) Map used by [Oliveira et al. 2015]

(b) Map used in this paper

Figure 1. Maps considered in this study

3. Greedy and Hungarian Algorithms

The Greed Algorithms (GA) presents, in this context, an instant response because, different of the other methods, it does not need to wait multiple calls to be executed. When the first call is made in the simulation, it is chosen a free vehicle (who was not answering any previous calls) with the shortest distance (based on Global Position System - GPS) from the call to answer it. On the other hand, the Hungarian Algorithm (HA) uses a call

[1]http://sumo.sourceforge.net/

buffer, which stores the outgoing calls for a short time and then processes the optimization model to identify the best vehicles to answer all calls. Thus, HA is able to search for the combination of vehicles / users which minimizes the total required distance to be traveled for each vehicle to find its respective customer.

4. Genetic Algorithm

To implement the Genetic Algorithm (GE), first it was defined how each possible solution should be coded as an individual. Then, how to select, cross and mutate these individuals in order to obtain the best result. All these processes and parameters used will be described in the next subsections. The GE used was based on the version proposed in [Stoltz 2018].

4.1. Individuals

The best way found to represent individuals was to numerate every autonomous vehicle (AV_0, AV_1, AV_2, ..., AV_m) and, for every call, to create an array of n-elements (where n is the number of free AVs). Each value contained in the array indicates which call the correspondent AV is designated to attend (value 5 on position 3 means that the third AV (AV_2) is answering call number 5). Following the array representation showed in Figure 2, the position 0 of the array indicates that the first AV (AV_0) was not answering any calls, but the second AV (AV_1) was selected to answer the second call. Those individuals were randomly generated. To improve the performance of the GE, a individual that represents the solution obtained by the GA is generated an included in the population at the beginning of the GE execution. An important thing to notice is that there are three types of AVs: (1) the AVs which were not answering any calls and will not receive a call to answer after the execution; (2) the AVs which will receive a call to attend after the execution, and (3) the AVs which were already attending calls, these last ones are not represented in the individuals array.

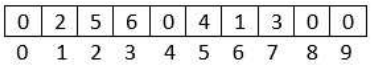


Figure 2. Visual representation of an individual, where there are 4 free AVs and 6 AVs answering calls

4.2. Population and Niche

At first, all the individual were put in a population, then to prevent the GE to converge to a local minimum, they were divided in 10 niches [Goldberg and Holland 1988, Pereira et al. 2014] which consist of sub-populations where the crossover happen between individuals in the same niche and at every few generations, one individual of every niche was selected to migrate to another niche. In the last offspring, all niches are concatenated to obtain the best solutions as a whole.

4.3. Fitness

To evaluate the individuals and sort them, a good fitness method was necessary. The simulation data provided by SUMO was used to create a table (fitness table) containing

all the AVs distances to all calls. The lines of the fitness table represents the AVs and the columns represents the distance to the respective calls (first column indicates the distance of all the AVs to the first call). Since every individual is composed of genes, each gene represents an AV and the call it is answering (Figure 2). To obtain the fitness value, it was only necessary to use the genes' values as the table indexes and find the distance of the selected AVs: taking Figure 2 as an example, AV_1 is answering the second call, the distance of that AV to the location of the call it is attending can be found on the second line and second column of the fitness table. After every distance is obtained, they are added, giving the fitness value of that individual. The niche fitness value is the mean of all individual's fitness values. This process was repeated for every niche. Then, it was generated a second array for every niche (fitness array), containing every individual's fitness value of those niches. In other words, the fitness array is an auxiliary array to store the fitness value of the individuals, avoiding the need to recalculate the fitness every time an individual needs to be added to a niche.

4.4. Sorting Method

One of the most time consuming methods was the Sorting. The solution found to improve the execution time of the GE was to avoid using the sorting method so many times (The first version of the GE used the sorting method every time after every offspring generation). Thus, all the individuals were sorted into its niches one time at the first generation and one more time at the last generation when the niches were merged into the whole population. To add the children and the mutated individuals to the niches without having to sort them again, the fitness of those new individuals were evaluated, then using the fitness array of the niches (see Section 4.3), those new individuals were put in the correct position, keeping the population in the niche ordered, without having to sort all individuals again. Individuals with the worst fitness value were removed, to simulate the evolutionary process and to maintain the niches size constant.

4.5. Selection and Crossover Methods

The Tournament selection method [Goldberg and Holland 1988, Stoltz 2018] was used, where a group of individuals are chosen randomly and that one with best fitness is selected as the first parent; then the process is repeated for the second parent. After the parents selection, the crossover process starts: each gene of the first parent is compared with the correspondent gene of the second parent and the one with better fitness (shortest distance) is chosen as the child's gene (as a survival of the fittest). In this method its possible that some calls do not be answered. To solve this issue, it was add a function that, for every unanswered call, a free AV with the shortest distance would be selected to attend them.

4.6. Mutation

There was implemented a bit-by-bit mutation [Vasconcelos et al. 2001], due to the size of individuals. Thus, there was two kinds of mutation rate: the first one was the probability of the individual participates of the mutating process and the second one was the chance of mutate each gene. Before selecting the gene to mutate, it was first decided if the mutation would affect free or occupied AVs (50% chance); if a free AV mutation was chosen, it was always changed with a occupied AV (since selecting another free AV would not change the final result). If a occupied AV was chosen, it could be changed with another occupied AV or with a free AV (both chosen at random).

4.7. Migration

As presented in Subsection 4.2, the population was divided in 10 niches and, after a few generations, the migration was performed. The migration consisted of selecting one individual of every niche, then choosing randomly another niche as a destination for migration. The process of adding it to the new niche followed the same procedures described in Subsection 4.4, which means that it is possible that a individual who were migrated could be reject by that niche, if its fitness value was worst than every other individuals' fitness values.

4.8. Parameters

The parameters used in this experiment were the following: population size of 1,000 individuals, 200 generations, crossover rate of 80%, mutation rate of 5%, gene-mutation rate of 0.5%, and the migration was performed at every 8 generations. It was made 250 vehicle's calls and there were 496 free AVs.

5. Results and Comparisons

The algorithms were executed on a Acer Aspire Nitro 5 AN515-51, with Intel® Core™ i7-7700HQ, 16 GB of RAM. To compare the results, the GE was executed 30 times for every test, due to its random nature. The GA and HA were executed only once for each test, since they give always the same expected answer. To compare the results the GE was executed 30 times for every test, due to its random nature. The GA and HA were executed only once for each test, since they give always the same expected answer. Than, 30 different tests were executed, changing the location and time of the calls. Those results were organized in Table 1.

Table 1. Comparison between the 3 methods presented on this paper

	Execution Time	Average Total Cost (Km)	Maximum Total Cost (Km)	Minimum Total Cost (Km)
Greedy Algorithm	Less than 1 second	296916	341487	273197
Hungarian Algorithm	About 40 minutes	272386	304838	248302
Genetic Algorithm	About 12 minutes	291331	339702	264816

The Genetic Algorithm showed a better answer of that obtained by the Greedy Algorithm (best case was about 3% better), but is still worse than the result obtained by the Hungarian algorithm, which is about 11% better than the GA. On the other hand, the GE ran faster than HA, as can be seen on the second column of Table 1.

6. Conclusion

The designation of vehicles in a complex map, as occurs in medium and large cities can be a challenge, especially with a high demand for vehicles, coupled with a busy traffic, where the choice of the best route is no longer trivial. This paper proposed a Genetic Algorithm that tries to obtain a good result, as can be reached by the Hungarian Algorithm, but in a

short time, as can be reached by a Greedy Algorithm. This combination is not trivial and needs to be improved with new configurations of the proposed algorithm, allied to new experiments, exercising different scenarios of demand and traffic. However, the results showed that the study is promising because the Hungarian Algorithm is not viable in a complex map and the results of the Genetic Algorithm are already superior to those obtained by the Greedy Algorithm, even though not quite as fast. A fine tune of the parameters used could lead to a faster execution without worsening the results obtained.

Acknowledgment

The authors thank CNPq and PROPE/UFSJ for financial support.

References

Behrisch, M., Bieker-Walz, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo-simulation of urban mobility: An overview. *The Third International Conference on Advances in System Simulation*, pages 63–68.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT press.

Goldberg, D. E. and Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

Jonker, R. and Volgenant, T. (1986). Improving the hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Oliveira, A. A. M., Souza, M. P., Pereira, M. A., Reis, F. A. L., Almeida, P. E. M., Silva, E. J., and Crepalde, D. S. (2015). Optimization of taxi cabs assignment in geographical location-based systems. *Proceedings XVI GEOINFO*, 1:92–104.

Pereira, M. A., Davis-Júnior, C. A., Carrano, E. G., and de Vasconcelos, J. A. (2014). A niching genetic programming-based multi-objective algorithm for hybrid data classification. *Neurocomputing*, 133:342 – 357.

Reis, F. A. L., Pereira, M. A., and Almeida, P. E. M. (2013). Location-based service to reduce the waiting time for taxi services. *Fourth International Workshop on Knowledge Discovery, Knowledge Management and Decision Support*.

Souza, M. P., Marinho Oliveira, A. A., Pereira, M. A., Lima Reis, F. A., Maciel Almeida, P. E., Silva, E. J., and Silva Crepalde, D. (2016). Optimization of taxi cabs assignment using a geographical location-based system in distinct offer and demand scenarios. *Revista Brasileira de Cartografia*, 68(6).

Stoltz, E. (2018). Evolution of a salesman: A complete genetic algorithm tutorial for python. *Towards Data Science*.

Vasconcelos, J. A., Ramirez, J. A., Takahashi, R. H. C., and Saldanha, R. R. (2001). Improvements in genetic algorithms. *IEEE Transactions on magnetics*, 37(5):3414–3417.