Road Traffic Simulation Regulated by Normal Traffic Lights and Green Waves

Maria Merlino

May 3, 2018

Abstract

The aim of the paper is to simulate both road traffic in an urban area and the phenomena related to the presence of crossings and traffic lights. In particular, I am going to study which parameters should be preferred in the traffic light systems to reduce traffic jams with greater flow. At last, I am going to compare the results with a different model of traffic lights: the green wave.

Contents

Introduction

In urban areas, roads are classified, from the Italian "highway code", as follows:

- *strada urbana di scorrimento*: road with independent carriageways or separated by traffic dividers, each with at least two lanes and a possible lane reserved for public transportation, and sidewalks to the right; there are special side areas behind the carriageway for parking. Any intersections are regulated with traffic lights.
- *strada urbana di quartiere*: single carriageway with at least two lanes, and sidewalks; beyond the carriageway there are areas for parking.

In the paper I focus on the typology *strada urbana di scorrimento*, simulating what happens in crossings with smaller urban roads.
Using NetLogo I have built a road model in which a certain number of vehicles move around across intersecting streets and changing lanes when they encounter slower cars. Every intersection is regulated by classic three-stroke traffic lights, however the Italian road code does not provide specific times for the duration of the colors, as they depend on traffic characteristics and therefore also vary within the same urban area. For this reason, the first part of the work is mainly dedicated to the analysis of the optimal traffic lights factors, using genetic algorithms. The GA are procedures aimed at solving research and optimization problems, using them I expect solutions that reduce the flow of machines at intersections or waiting times at traffic lights.
Later on I modified the model by inserting different traffic light systems: green wave traffic lights. They are type of synchronized electronic traffic lights, which allow the driver of the car to cover a section of road with multiple intersections always finding the green lights, provided he follows the speed indicated. I intend to show that the use of green wave traffic lights reduces waiting times and generates a smoother flow of traffic.

NetLogo

To simulate the traffic I used NetLogo. It is a agent programmable modeling environment that enables exploration of emergent phenomena.
NetLogo provides a complete simulation environment, allows both to implement a model and to create a graphical user interface; also it allows exploration by modifying switches, sliders, choosers, inputs, and other interface elements. NetLogo allows authoring of new models and modification of existing models, many of these can be found in the NetLogo library or in the community.
For my work I modified the *Town - Traffic and Crowd simulation* model by Lukas from the community (see [4] for more details). The model generates a main road that crosses secondary roads and simulates an interaction between three kinds of agents: traffic lights, vehicles and pedestrians. Each car can accelerate, decelerate, change lanes and turn left and right. The vehicles can not exceed the speed limit but they will try to overtake the slower cars. Pedestrians can go wherever they want, cross the road by using wait point and crossings. If there are any other pedestrians in front of them, they turn slightly and wait.
The graphic interface allows you to interact with the model and set some parameters: number of pedestrians and cars, lights interval, acceleration and decelerate of the cars, road speed limit, probability of turning, time to crossing and basic politeness (sets the value for computing of politeness of cars). There is also a switcher for the turn left or not.

I made some changes to the model. I have reduced the size of the sidewalks and changed the size of the NetLogo *world*. In this way at least three intersections are clearly visible.
In the Lukas' model, the traffic lights have only green and red light, therefore, I have modified the code and inserted the yellow light (the machines slow down when they see it). I canceled the *lights-interval* button and inserted three new buttons, one for each light of the traffic light. The new buttons help to achieve the objective of this paper that is, to study the setting of the traffic lights to reduce road traffic.
I wanted to make the simulation more real so I changed the synchronization of the traffic lights; in the new model each green light is preceded by a red

one and each red light is preceded by a green one. Finally, I added the plot to know the number of cars waiting at the lights, useful for setting the traffic light intervals. (Figure 2.1 and 2.2)

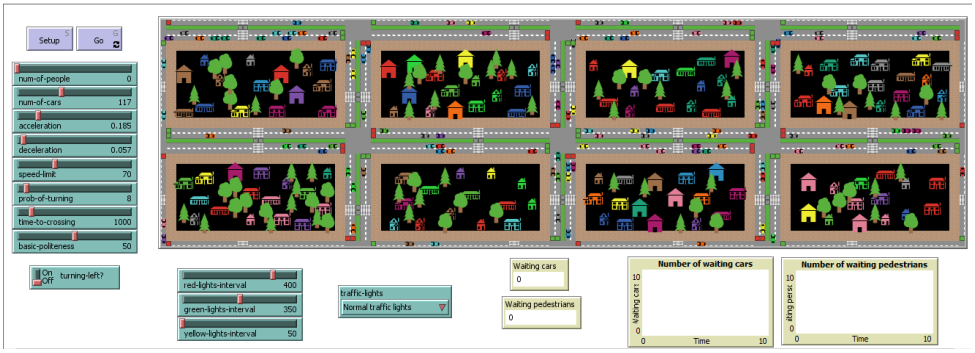
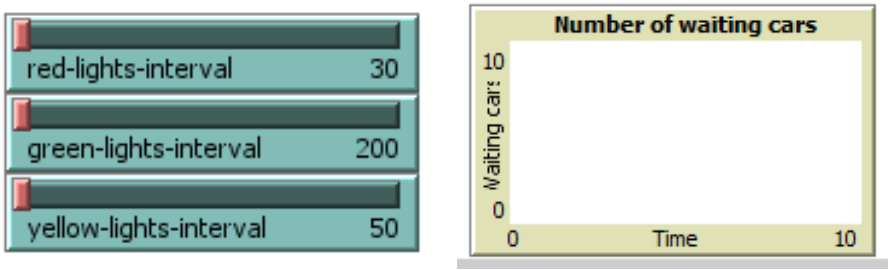


Figure 2.1: Graphical user interface of the model

2.1



(a) *lights-interval.* **(b)** *plot for waiting machines.*

Figure 2.2: some parameters

2.2

Normal traffic lights

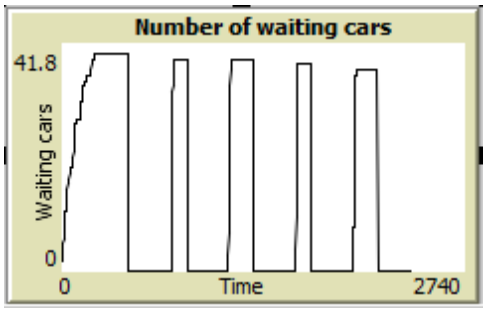
In the NetLogo graphical interface, I created a *Chooser* that allows you to choose between two simulation modes: normal traffic lights and green wave. *Normal traffic lights* indicates traffic lights alternating with a standard color. The green light allows the use of the other side of the intersection and the yellow light warns that the signal is about to change to red. The yellow light is a phase in which drivers must slow down and they can cross the intersection only if safe to do so. The red light is obviously a stop sign.
The traffic light intervals can be changed through the *sliders*, but I have imposed a condition: the minimum value of the red light interval must be greater than or equal to the sum of the other two; only in this way, at the same intersection, there are no cases in which the traffic lights are all green. I have also set up that minimum and maximum of yellow lights are about a quarter of the corresponding green values. In this way, during the simulation, the yellow light has short times but we can also notice what happens when the machines find it.
The cars have a random speed lower than the speed limit and can go where they prefer, but they are obliged to respect the highway code. They can decide whether to give priority to pedestrians waiting on the crosswalk.

3.1 Results

During the simulation, if there are few machines, we do not see traffic jams or long lines of cars but the machines spend a lot of time standing at the traffic light. When the number of machines reaches the order of hundreds, the situation changes. In particular, near the red traffic lights, we look very large *blocks* of machines (figure 3.1).
This type of road flow forces drivers to accelerate and decelerate very often. As a consequence, car consumption and smog increase. There are long lines of cars stopped at traffic lights and waiting times are significant, especially if the traffic lights are not set appropriately. (Figure 3.2)
The example in the figure 3.2 shows the results of a simulation with random

Figure 3.1: Almost all the cars are stopped and waiting for the green light.

3.1



3.2

Figure 3.2: The graph shows the number of machines stopped due to the red light. The function has a stepped pattern, which reflects the *blocks* of machines we see in the simulation.

parameters. If we change the parameters of the traffic light intervals, the pattern does not change much, while the waiting times decrease or increase. At this point, I used Behavior search to optimize the results. It is a software tool to help with automating the exploration of agent-based models, by using genetic algorithms and other heuristic techniques to search the parameter-space.

Genetic Algorithm

Genetic algorithms are research algorithms inspired by the process of natural selection and evolution. In particular, they allow to analyze and recombine the starting hypotheses to optimize the solution of a problem. They are algorithms useful for the exploration of complex simulation models but they do not always find an optimal solution.
The algorithm starts with a population of candidate solutions to an optimization problem. The solutions are called phenotypes or creatures and each one has a set of properties called genotype or chromosomes which can be mutated and altered. The starting individuals are random and with each solution the algorithm associates a degree of evaluation using the fitness function. The evolution is an iterative process, in each iteration the population called a generation. The more fit individuals are stochastically selected from the current population, and each individual's genome is recombined to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. In some cases the evolution reaches the optimal conditions and stops, to avoid this problem the algorithm introduces occasional mutations.
Genetic algorithms find application in bioinformatics, computational science, engineering, economics, chemistry, mathematics, physics and other fields. The GA are simple to implement, even if their behavior is complex.
In computer science, to understand the genetic algorithms we must speak first of all of the search algorithms. They are algorithms that look for an object with specific properties between a collection of objects. The search for algorithms consists of a series of rules to be executed each time the described conditions are met and during the search, the algorithms must also respect the constraints of the problem.
The coding of the problem occurs in different ways. A standard representation use the binary code: the conditions and actions are represented by bit strings corresponding to the presence or absence of specific characteristics in the input and output of the rules. Each candidate solution is as an array of bits, 0 and 1 identify the characteristics of the solutions. For Genetic Algorithms, it is equivalent to constructing a *genetic code* that represents the structure of the problem, just like DNA with living organisms.
In every generation, the estimate of each individual in the population is

evaluated and eventually we get fitness, usually it is the value of the objective function in the optimization problem to be solved. Each candidate solution is a series of bits. During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions are typically more likely to be selected. The algorithm ends when or has been produced a maximum number of generations, or satisfactory the level of security has been reached for the population.
Typically, a genetic algorithm provides a genetic representation of the solution domain and a fitness function to evaluate the solution domain. The fitness function is defined over the genetic representation and measures the quality of the represented solution.

BehaviorSearch

BehaviorSearch aims to facilitate model analysis by making search and optimization techniques accessible to all modelers and it interfaces with NetLogo, to provide a low-threshold way to search for combinations of model parameter settings that will result in a specified target behavior (see [5] for more details).
Model exploration works through four steps:

- Design a quantitative measure for the behavior you're interested in.
- Choose parameters to vary and what ranges are allowed.
- Choose a search algorithm and run it.
- Examine the results.

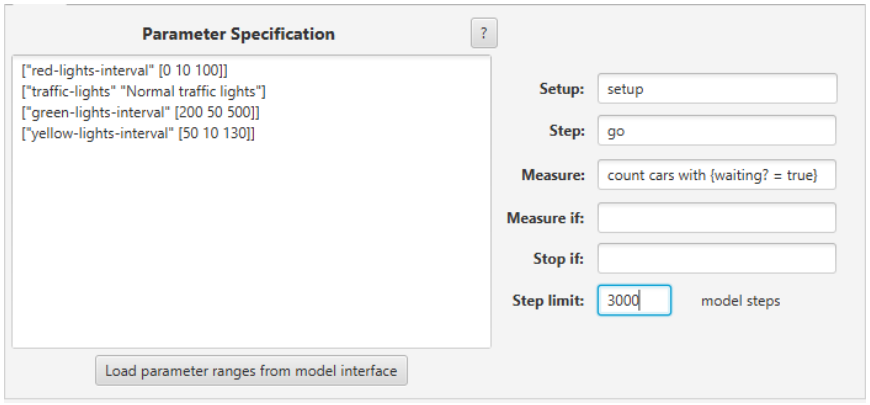
I decided to use BehaviorSearch to improve the results obtained in previous simulations. The goal is to reduce the number of machines waiting, so the genetic algorithm must modify the parameters: *red-lights-interval*, *green-lights-interval* and *yellow-lights-interval*.
The figure 5.1 shows the parameters chosen to start the program.

5.1 Results

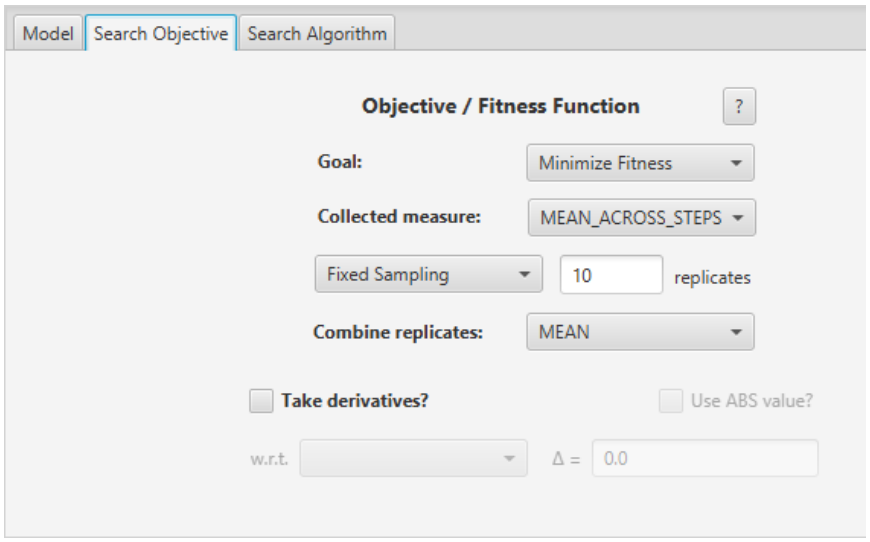
BehaviorSearch allows you to do many tests with different parameters, I decided to report only the test that generated the best fitness (see figure 5.2). The program generated the fitness in the figure using the following values:

- *red-lights-interval*: 30;
- *green-lights-interval*: 250;
- *yellow-lights-interval*: 50.

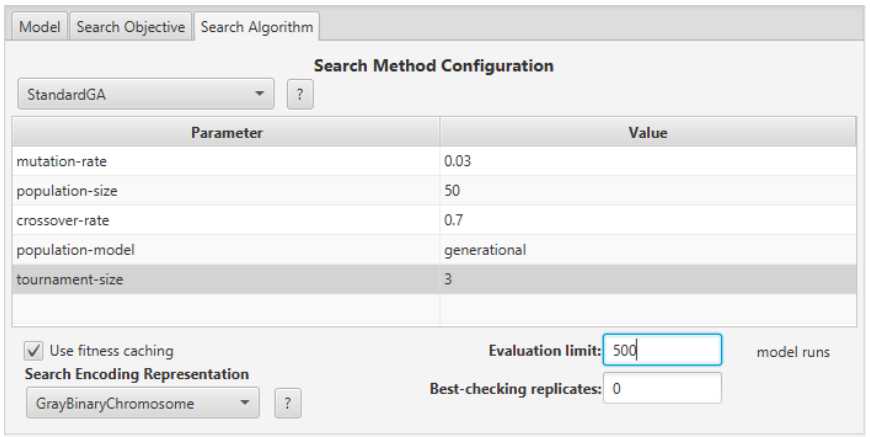
Using the parameters obtained with BehaviorSearch, the results of the simulation on NetLogo change (see figure 5.3). Wait times have decreased significantly and the number of machines waiting is slightly fluctuating.
We can see that the situation has improved compared to the original case. However, the road flow is not fluid enough to avoid the formation of long lines of stationary vehicles.



(a) *Model.*



(b) *Search Objective.*



(c) *Search Algorith.*

Figure 5.1: The settings chosen to BehaviorSearch

5.1

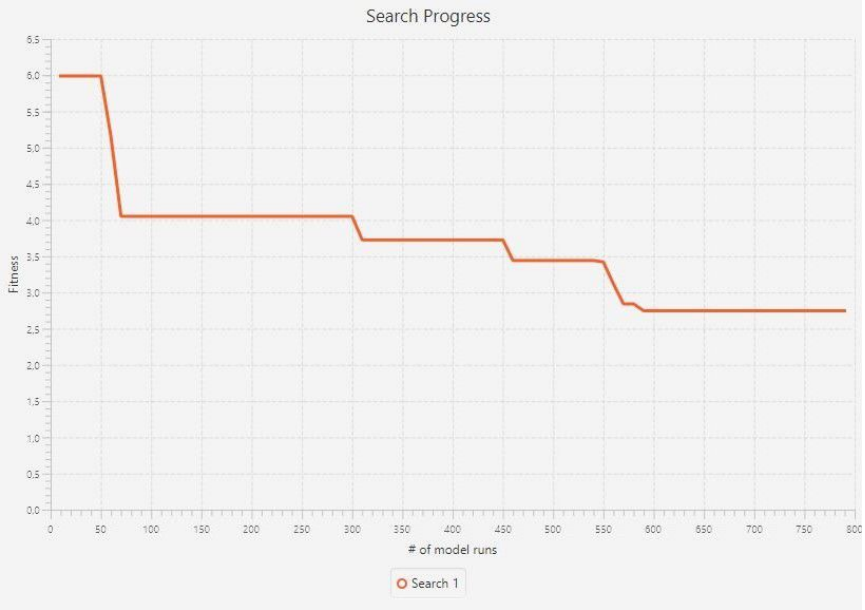


Figure 5.2: The fitness.

5.2

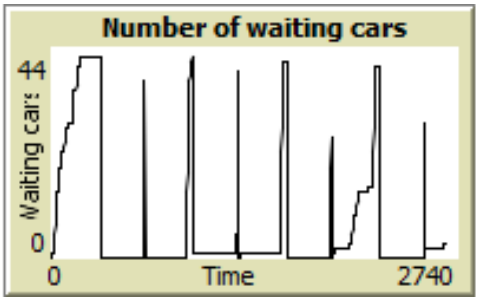


Figure 5.3: Plot of the waiting machines with the new parameters.

5.3

Green wave

Green wave indicates a series of traffic lights coordinated to allow continuous traffic flow over several intersections in one main direction. The cars traveling along with the green wave will see a progressive cascade of green lights, and not have to stop at intersections. In practical use, only a group of cars (known as a *platoon*, the size of which is defined by the signal times) can use the green wave before the time band is interrupted to give way to other traffic flows. In conventional setups only the roads and directions with the heaviest loads get this preferential treatment.

In the model, the green wave traffic lights are configured in the horizontal central road, it is the main road. Each traffic light on this road indicates to vehicles the speed to enter the green wave flow. The machines assume a speed close to that indicated but not identical (through the *random-normal* command), in this way the simulation is more realistic. For the simulation I chose to change the speed by 15% compared to that indicated. Of course, if I take a minor variation, I get better results.

6.1 results

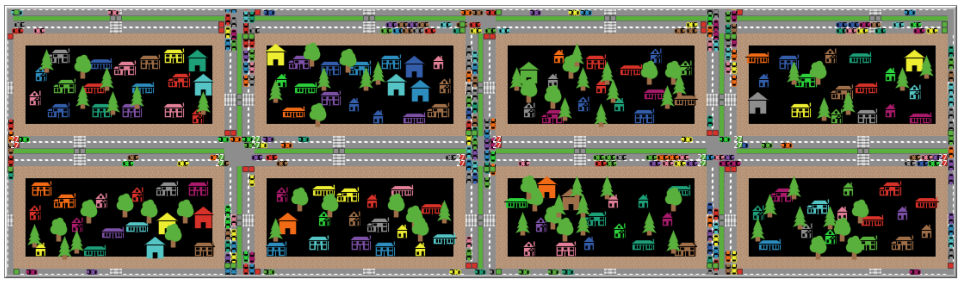


Figure 6.1: Simulation with green wave.
6.1

During the simulation, on the main road, we can see the changes (figure 6.1). Traffic flows more smoothly and there are no long lines of cars stopped at the red light. Therefore, waiting machines are decreasing and

there are fewer traffic jams.
I made the simulation with the values obtained with BehaviorSearch and the figure 6.2 shows the results. Also the number of cars is equal to the case of the normal traffic light (about 200).
In both cases, normal traffic light and green wave, the count of the cars

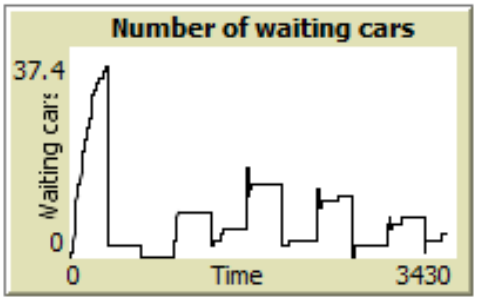


Figure 6.2: The graph shows the number of machines stopped due to the red light. The first peak of the function is higher than the others because the system has yet to stabilize.

6.2 stopped at the lights was done only on the main road.

Conclusions

The goal was to study road traffic on a *strada a scorrimento veloce* and reduce the problem of traffic jams.
The problem was faced with two types of traffic lights and the analysis of the results shows that the goal has been achieved.
The green wave traffic light makes the road traffic smoother and reduces the accelerations and decelerations of the cars (which consequently produce less smog). However, this is a solution that can not be applied to all roads. This type of traffic light involves only roads with straight corners. Furthermore, it is very important that most vehicles comply with the indicated speed, otherwise the benefits are minimal. A possible continuation of the work could be the introduction of the intelligent traffic light. The intelligent traffic light is a special type of traffic light whose purpose is to keep the cycle time on the main roads higher to favor speed and profitability and is more widely applied to crossings of roads with different importance and flow.

Bibliography

[1] *BehaviorSearch*, web site. 2018. *http* : *//www.behaviorsearch.org/*

[2] *Genetic algorithm* on Wikipedia. 2018.
https : *//en.wikipedia.org/wiki/Genetic_algorithm*

[3] *Green Wave* on Wikipedia. 2018.
https : *//en.wikipedia.org/wiki/Green_wave*

[4] Lucas, Jiri (2014), *Town - Traffic & Crowd simulation.* NetLogo User Community Models.
http : *//ccl.northwestern.edu/netlogo/models/community/Town*%20−%20*Traffic*%20&%20*Crowd*%20*simulation*

[5] Massobrio, Gerson (2012), *Optimizing the behavior of trading agents using genetic algorithms in a stock exchange simulation framework with real data. http* : *//terna.to.it/tesi/massobrio.pdf*

[6] Miller, John H. (1998), *Active Nonlinear Tests (ANTs) of Complex Simulation Models.* Management Science 44(6):820-830.

[7] *NetLogo User Community Models*, web site. 2018.
http : *//ccl.northwestern.edu/netlogo/models/community/index.cgi*