Proactive Control of Traffic in Smart Cities

Benjamín Zavala and Germán H. Alférez
Global Software Lab
Facultad de Ingeniería y Tecnología, Universidad de Montemorelos
Montemorelos, N.L., Mexico

Abstract *- The excessive growth of modern cities generates major problems in public administration. One problem is the control of traffic flow during peak hours. In this paper we propose a solution to the problem of vehicular control through a proactive approach based on Machine Learning. Through our solution, a traffic control system learns about the traffic flow in order to prevent future problems of long queues waiting at traffic lights. The traffic system architecture is based on the principles of Autonomic Computing to change the timers of the lights automatically. A simulation of the streets on a smart city and a tool based on Weka were created in order to validate our approach.*

Keywords: Machine Learning, Proactive Control, Traffic, Smart Cities, Autonomic Computing.

1 Introduction

In recent years, the world has experienced mass migration from the countryside to the city. For example, in Mexico today over 70% of the population lives in the city [1]. There is an exponential growth in the population of cities turning them into mega cities. This phenomenon leads to many problems. One of these problems is traffic congestions [2]. Traffic congestions negatively affect the quality of life of citizens by increasing travel time, generating stress and economic losses, and increasing environmental pollution.

The current solutions to control traffic in smart cities wait for an event to happen (i.e., a long queue at a traffic light) to generate an action to try to solve this event. These solutions may be referred to as "reactive". However, we believe that for cities to become truly intelligent, they need "proactive" solutions that anticipate traffic problems and prevent these problems from becoming evident.

In this paper, we present a proactive solution for controlling vehicular traffic in smart cities. Our solution is based on Machine Learning and Autonomic Computing. First, our approach analyzes data traffic levels with Machine Learning. Using this technique of artificial intelligence, the system takes proactive decisions based on historical traffic data. In order to perform autonomous adjustments at traffic lights, the system self-adjusts according to the principles of Autonomic Computing of IBM [3]. A simulation of traffic in a smart city demonstrates the efficiency of our solution. In order to predict the problems that can occur in the future, our solution is based on Weka API [4].

The remainder of this paper is organized as follows. Section II presents a simulation that demonstrates the problem of traffic congestions in big cities. Section III presents related work. Section IV presents the underpinnings of our approach. Section V describes our solution to carry out proactive adaptations of traffic lights. Section VI presents our running prototype and evaluation results. Section VII presents conclusions and future work.

2 Traffic congestions in big cities

In this section we present a simulation of the streets of a city that helps to understand the arising traffic problem. This simulation consists of a vehicular crossing of two unidirectional streets. In order to simplify the simulation, only the change between green and red light signals is simulated. Congestion becomes evident when the amount of vehicles that can cross a traffic light in a particular moment is lower than the amount of arriving cars to the queue.

In the simulation, every traffic light has an assigned time value "x" for red or green. Also, we specified the average time that a car takes to cross an intersection ("y"). Based on these data we calculated how many vehicles could cross the traffic light when it is green. Also, a random value was generated for the vehicles that come to the queue in the traffic light "i".

As seen in Figure 1, the number of vehicles tends to increase linearly as time passes. In our simulation, this trend was generated when "x" takes a value equal to 15 seconds, "i" has values between 0 and 9, and "y" equals to 3 seconds.

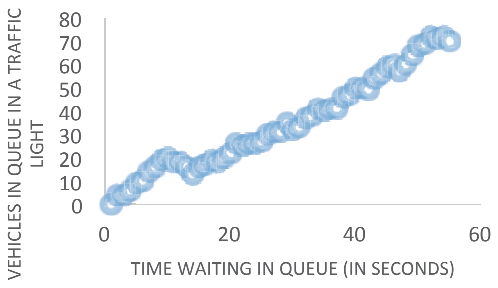


Figure 1. Results of the traffic simulation

3 Related work

Currently fixed-time and sensitive traffic strategies have been implemented for managing traffic flow. These strategies are presented below.

3.1 Fixed time strategies

Fixed time strategies are adjusted for long periods of time where parameters are assumed constant. This approach can be problematic in settings with high variability demands or with usual presence of non-standard conditions (e.g. accidents, riots, or unexpected events).

For example, SIGSET is based on traffic flow patterns at an intersection traffic light. This is a well-known system for traffic engineers [5]. SIGSET works in isolation at each intersection and assigns fixed times at traffic lights.

3.2 Sensitive traffic strategies

Sensitive traffic control strategies execute their logic based on traffic measurements performed in real time at the entrances of the intersections. In order to perform measurements, it is necessary to have some type of traffic detectors.

Within traffic sensitive methods, there are two reactive methods that solve problems when they are already evident. On one hand, there are approaches that detect the presence of heavy traffic at an intersection and change traffic light timers to give preference to the direction of greatest activity. On the other hand, there are adaptive solutions with networks of traffic lights. Plans are implemented together in these networks in order to optimize traffic flow [6].

Another sensitive strategy is a proactive one. For instance, RHODES [7] takes as input data of real-time measurement of traffic flow. Then, it controls the flow through a network. The system uses a control architecture that decomposes the traffic control problem into several sub-problems that are interconnected in a hierarchical fashion and predicts traffic flows at appropriate resolution levels (individual vehicles and platoons). This approach has several optimization modules for solving hierarchical sub-problems. Also, it uses a data structure and communication approaches to reach fast solutions of sub-problems. RHODES depends on a central control module. We argue that a more decentralize approach could be used to distribute calculations on site. In this way, the costs and complexity related to infrastructure communication could be reduced.

4 Underpinnings of our approach

The solution proposed in this paper is intelligent, autonomous, and proactive. These underlying concepts are described below.

4.1 Machine Learning

Machine Learning is a term used to encompass a wide variety of techniques for discovering patterns and relationships in sets of data. The primary goal of any Machine Learning algorithm is to discover meaningful relationships in a set of training data and produce a generalization of these relationships that can be used to interpret new, unseen data [8]. Among Machine Learning methods, we can find forecasting. Forecasting is the process of making statements about events whose actual outcomes have not yet been observed [9].

4.2 Autonomic Computing

Inspired by biology, Autonomic Computing has evolved as a discipline to create self-managing software to overcome the complexities to maintain systems effectively. Autonomic Computing covers the broad spectrum of computing in domains as diverse as mobile devices and home-automation, thereby demonstrating its feasibility and value by automating tasks such as installation, healing, and updating. Since doing manual adaptations is a difficult (and sometimes impossible) task, our approach is based on IBM’s reference model for autonomic control loops (which is sometimes called the MAPE-K loop) [10].

4.3 Proactive Adaptations

On one hand, reactive adaptations are performed in response to an event. On the other hand, proactive adaptations take an action in advance (i.e., before an incident negatively impacts the system) [11]. Reactive adaptation mechanisms may cause increases in the execution time and financial loss, which can lead to user and business dissatisfaction [12]. Proactive approaches try to solve these problems by detecting the need for an adaptation before the problem is evident.

5 Our approach

In this section we present our solution to carry out proactive adaptation of traffic lights. The structural blocks of our approach are based on the components of the MAPE-K loop, i.e., Monitor, Analyze, Plan, Execute, and Knowledge (see Figure 2).

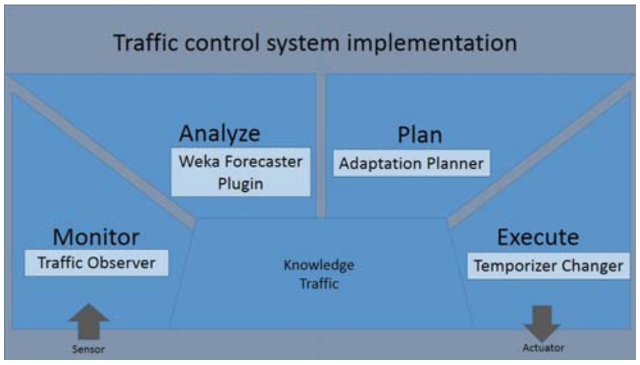


Figure 2. MAPE-K cycle of our approach

In our approach, sensors collect traffic data in the Analysis component. This data is used for training our Machine Learning approach. The task of observing the traffic is in charge of the Traffic Observer, which detects violations of any Service Level Agreement (SLA) at specific times. A violation of the SLA becomes evident when the number of cars in queue is greater than the desired SLA. After completing training, the Weka Forecasting plugin analyzes the data collected and predicts potential traffic problems. Then, the Adaptation Planner plans the necessary changes to the traffic light timers in order to prevent traffic problems proactively.

Finally, the Temporizer Changer makes the necessary changes in the traffic light timers trough actuators.

Our solution is described in the following subsections based on the MAPE-K components. Our approach is exemplified based on the traffic simulation described in Section II.

5.1 Monitor

Monitoring involves capturing properties of the context that are meaningful to the self-properties of the system. In our case, we are interested in observing the traffic. To this end, we propose the Traffic Observer, which is a tool that observes the traffic through sensors. In our case, the Traffic Observer periodically checks the activity in the traffic simulation described in Section II with simulated sensors.

At the start of the simulation, we create two traffic lights. A status, either green or red, was assigned to each one of them. Also, a random number of vehicles that come into the queue was generated. In order to simplify the simulation, the preventive stage of a traffic light, which is usually yellow, was included within the green phase.

The Traffic Observer detects if the number of vehicles queued at a certain moment at the traffic light is greater than the SLA (i.e., a violation of the SLA). If so, this event is saved in a log with .arff extension. Files with this extension have the required format to carry out forecasting in Weka. Specifically, the .arff file format requires that the data has a special header, such as the one in the following example:

```
@relation A @attribute seconds numeric
@attribute cars numeric @data 60, 7
```

First of all, there is a relationship of data (@relation A), and this relationship has two attributes, both of type numeric: seconds (@attribute seconds) and cars (@attribute cars). After this, the problematic event (i.e., a violation of the SLA) data is recorded (@data 60, 7). In the example above there is a violation of the SLA after 60 second of execution. At this time, there are 7 vehicles in queue, when the SLA indicates that the maximum number of cars in queue should be 3 cars.

5.2 Analysis

The objective of this phase is to detect, in a proactive manner, the traffic problems that may occur according to the results captured in the log generated by the Traffic Observer. In order to accomplish the prediction of traffic problems, we use the Weka Forecasting plugin. This plugin can load or import a time series forecasting model and use it to generate a forecast for future time steps beyond the end of incoming historical data [13]. Among the forecasting methods available in Weka, we chose the Multilayer Perceptron due to its lowest percentage of error after completing tests with different methods (Gaussian Process, Kernel Regression, Linear Regression, Multilayer Perceptron, and SMOreg).

In order to run the forecaster, the following parameters are required: type of data to predict, measure of time for training data, and the number of times to predict (e.g. the epochs). In our case, the forecaster predicts the number of vehicles that will arrive at the traffic light after training. The data generated by the forecaster is stored in a text file.

For example, a possible training (i.e., the observation of the traffic for a period of time) concluded in the second 1,800 of the simulation execution. During this time, 25 violations of the SLA were found. With this data, predictions are performed. Specifically, the forecaster predicts that if the traffic lights keep operating with the current value of their timers, then there will be a greater number of vehicles than those specified in the SLA in the second 1,860. Through forecasting, it is possible to predict problems ahead based on historical data.

5.3 Planning

The objective of this phase is to plan how to automatically solve the traffic problems predicted in the analysis phase. Specifically, in this stage the traffic light timers are recalculated in order to avoid traffic problems.

During planning, the file created at the analysis phase is read. Each entry in this file is a problem to solve (i.e., a violation of the SLA). In this phase, we propose the Adaptation Planner. This tool performs the following steps to plan a change in traffic light timers:

1. The Adaptation Planner keeps in a variable the text recovered from the file generated in the previous phase. For example, the Adaptation Planner takes the data of 35 vehicles in queue that will violate the SLA at some future time (after training) according to forecasting.

2. The Adaptation Planner performs the following operation:

   *newTimer = (int) (d * tCross);*

   The "d" variable is the number of cars that could violate the SLA according to forecasting. The "tCross" variable indicates the average time that a vehicle takes to pass the intersection. The "newTimer" variable is the new traffic light timer. The result of the operation is transformed to integer.

3. The value of the "newTimer" variable, which corresponds to the solution to a problem (i.e., a possible violation of the SLA), is saved. For example, a vehicle takes around 3 seconds to cross the intersection (tCross) and there are 35 expected vehicles that will arrive at the traffic light (d). Therefore, the result of *newTimer* is 105. It means that 35 vehicles take 105 seconds to cross the intersection.

Each of the values calculated in this phase corresponds to the solution of the problems referred to in the analysis phase.

5.4 Execution

The objective of this phase consists of making changes in the timers of the traffic lights according to the results of the planning phase. The idea in this phase is to prevent the materialization of predicted problems. Actuators are in charge of making changes in traffic light temporizers. In our case, actuators are simulated. These actuators take the new time for a traffic light ("newTimer" in the previous phase), and modify the traffic light timer with that time.

For example, the Adaptation Planner in the planning phase discovered that it takes 105 seconds for 35 vehicles to pass through the intersection. Therefore, at this stage our Temporizer Changer assigns 105 seconds to the timer of the traffic light. As a result, the 35 cars will have enough time to cross.

6 Running prototype

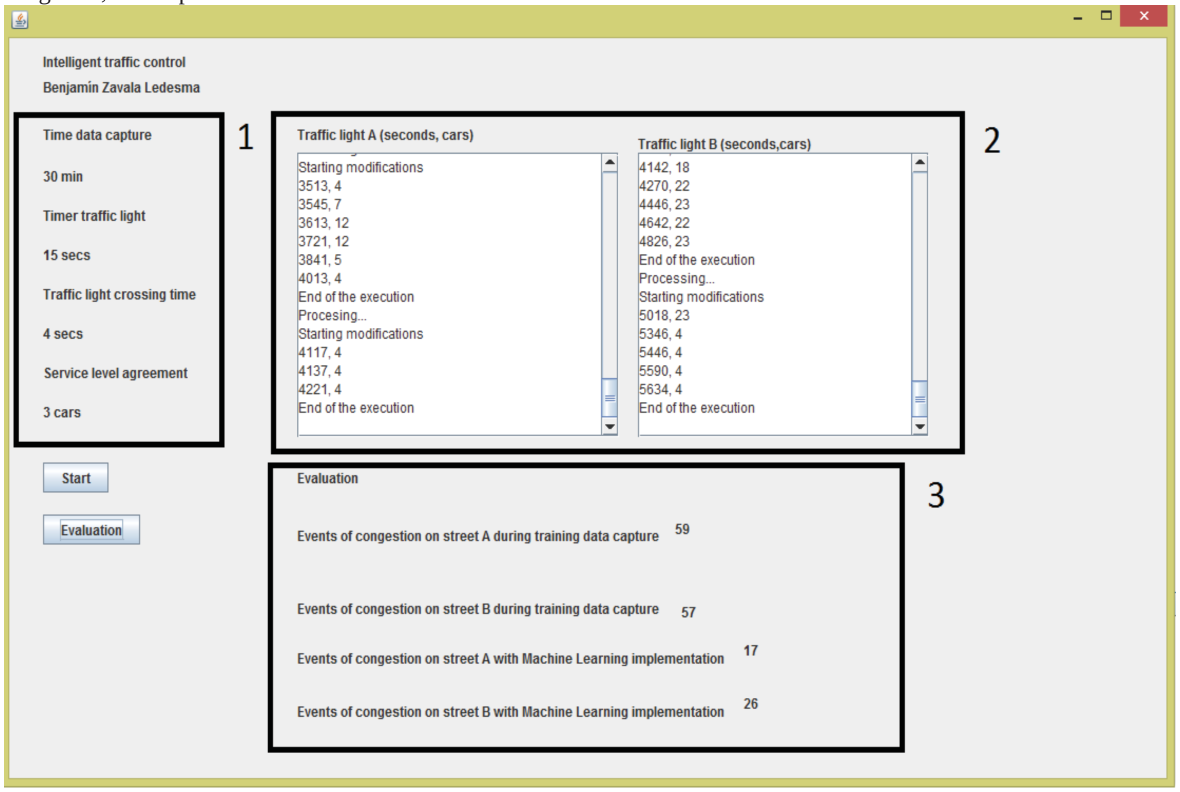
This section describes the prototype that has been created to demonstrate our approach. The prototype was created in Java and Weka, which can be accessed through Java [14].

The prototype GUI is divided into three areas (see Figure 3). The parameters under which the traffic control is performed are displayed in Area #1. Specifically, the following parameters are determined in this area: traffic light timer, execution time of the traffic simulation, crossing time, and expected SLA.

The text fields in Area #2 show the SLA violations in each traffic light (Traffic Light A and Traffic Light B). The first column indicates the time in seconds in which the violation of the SLA happened. The second column indicates the amount of cars with which the SLA was violated. Area #3 shows how many times the SLA at every traffic light is violated before and after the implementation of Machine Learning.

The prototype is trained according to the parameters in Area #1 of Figure 3. During training, the number of times that the SLA was violated is pretty high at each traffic light (e.g. 59 violations at the traffic light A and 57 violations at the traffic light B). Then, by using forecasting the number of SLA violations descends dramatically (17 violations at the traffic light A and 26 violations in traffic light B). Incidents of violation of the SLA are not fully eliminated but decrease greatly.

Figure 4 shows how Machine Learning dramatically decreased the SLA violations in our running prototype. The execution of Machine Learning occurs in the second 1,800. From this moment it is possible to see how the incidences of violations of SLAs are kept at a very low level.



Intelligent traffic control

Benjamin Zavala Ledesma

Time data capture

30 min

Timer traffic light

15 secs

Traffic light crossing time

4 secs

Service level agreement

3 cars

Traffic light A (seconds, cars)

```
Starting modifications
3513, 4
3545, 7
3613, 12
3721, 12
3841, 5
4013, 4
End of the execution
Procesing...
Starting modifications
4117, 4
4137, 4
4221, 4
End of the execution
```

Traffic light B (seconds,cars)

```
4142, 18
4270, 22
4446, 23
4642, 22
4826, 23
End of the execution
Processing...
Starting modifications
5018, 23
5346, 4
5446, 4
5590, 4
5634, 4
End of the execution
```

Evaluation

Events of congestion on street A during training data capture	59
Events of congestion on street B during training data capture	57
Events of congestion on street A with Machine Learning implementation	17
Events of congestion on street B with Machine Learning implementation	26

Figure 3. Implementation of the prototype

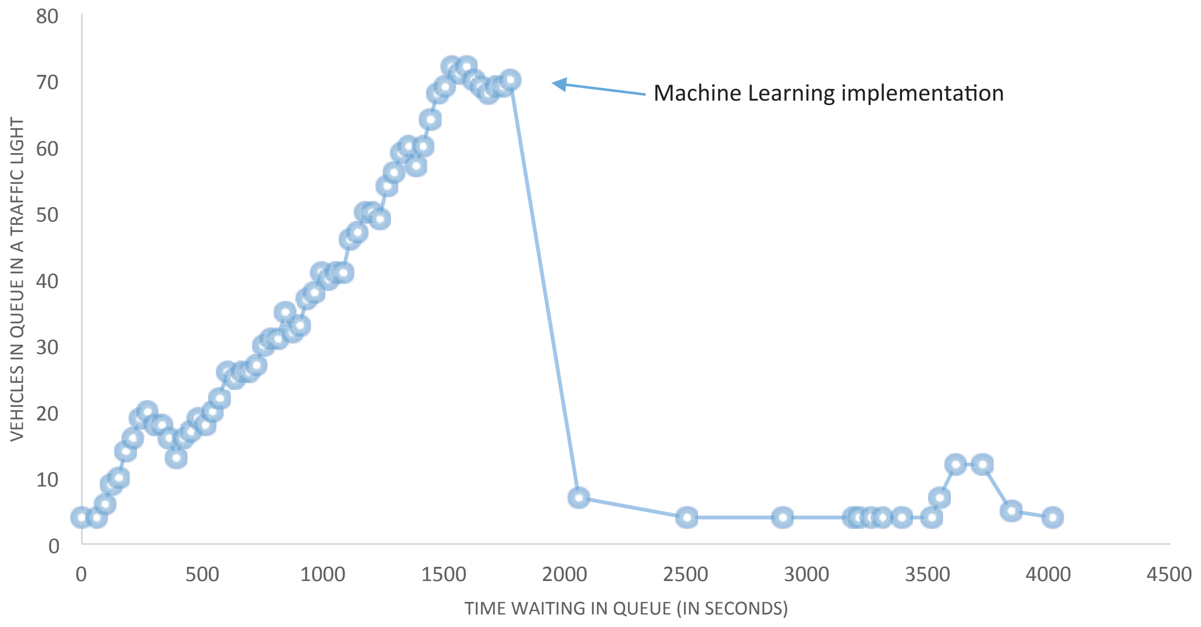


Figure 4. Application of Machine Learning in the second 1,800 of our traffic simulation

7 Conclusions and future work

This paper proposed a proactive solution to traffic control by means of Machine Learning and the principles of Autonomic Computing. Our solution could offer several benefits for the development of mega cities: 1) economic losses can be avoided by allowing that a cargo or a worker arrive in the shortest possible time to their destination; 2) pollution can be reduced; and 3) the number of vehicle accidents in rush hours can be reduced. These accidents may be caused by the stress produced in traffic jams [15].

As future work, we will implement a computer vision module for live traffic control through cameras. The Traffic Observer will collect this data. We will also develop a mobile application to help users make queries about the status of the traffic and find the most optimal path between the starting point and the destination point by getting data in real time from our autonomic solution.

8 References

[1] United Nations. (2015). UNFPA – Country programmes and related matters [Online]. Available: https://data.unfpa.org/docs/mex

[2] SIEMENS. (2015). *Siemens traffic solutions*. Available: https://www.swe.siemens.com/spain/web/es/industry/mobility/Documents/traffic.pdf

[3] IBM. (2006). "An architectural blueprint for autonomic computing," *IBM White Paper*.

[4] Pentaho. (2015). *Time Series Analysis and Forecasting with Weka*. Available: http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka

[5] R. E. Allsop. (1981). "Computer program SIGSET for calculating delay-minimissing traffic signal timings description and manual for users". University College London, Transport Studies Group.

[6] D. Robles, P. Ñañez, and N. Quijano. (2009). "Control y simulación de tráfico urbano en Colombia: Estado del arte," *Revista de ingeniería*, pp. 59-69.

[7] P. Mirchandani and L. Head. (2001). "A real-time traffic signal control system: architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, pp. 415-432.

[8] S. Mitchell. (1995). "The application of machine learning techniques to time-series data"

[9] J. S. Armstrong. (2001). *Principles of forecasting: a handbook for researchers and practitioners* vol. 30: Springer Science & Business Media.

[10] C. Ayora, V. Torres, V. Pelechano, and G. H. Alférez. (2012). "Applying CVL to business process variability management," in *Proceedings of the VARiability for You Workshop: Variability Modeling Made Useful for Everyone*, pp. 26-31.

[11] A. Metzger. (2011) "Towards accurate failure prediction for the proactive adaptation of service-oriented systems," in *Proceedings of the 8th workshop on Assurances for self-adaptive systems*.

[12] R. R. Aschoff and A. Zisman. (2012). "Proactive adaptation of service composition," presented at the Proceedings of the 7th International Symposium on

Software Engineering for Adaptive and Self-Managing Systems, Zurich, Switzerland.

[13] Pentaho. (2013). *Using the Weka Forecasting Plugin*. Available: http://wiki.pentaho.com/display/DATAMINING/Using+the+Weka+Forecasting+Plugin

[14] Pentaho. (2013). *Weka 3: Data Mining Software in Java*. Available: http://www.cs.waikato.ac.nz/ml/weka/

[15] Urbeconomica. (2015). "Accidentes vehiculares," vol. 2015, ed. Puebla: Redacción Urbeconómica.