Universidad de Montemorelos
Facultad de Ingeniería y Tecnología

CONTROL PROACTIVO DE TRÁFICO VEHICULAR MEDIANTE MACHINE LEARNING Y COMPUTACIÓN AUTÓNOMA

Informe de Investigación
presentado en cumplimiento parcial
de los requisitos para el grado
de Ingeniero en Sistemas
Computacionales

Por:

Benjamín Zavala Ledesma

Abril 2015

CONSTANCIA DE AUTORIA Y CESIÓN DE DERECHOS DE REPRODUCCIÓN

El abajo firmante, AUTOR del informe de investigación titulado CONTROL PROACTIVO DE TRÁFICO VEHICULAR MEDIANTE MACHINE LEARNING Y COMPUTACIÓN AUTÓNOMA por intermedio de la presente, DA FE de la autoría y originalidad de la obra mencionada que se presenta ante la Facultad de Ingeniería y Tecnología para ser evaluada con el fin de obtener el Grado Académico de Licenciada/o en Ingeniería en Sistemas Computacionales. Asimismo, dejo expresada mi conformidad de ceder los derechos de reproducción y circulación de esta obra, en forma NO EXCLUSIVA, a la Facultad de Ingeniería y Tecnología de la Universidad de Montemorelos. Dicha reproducción y circulación se podrá realizar, en una o varias veces, en cualquier soporte, para todo el mundo, con fines sociales, educativos y científicos. Entiendo que dicha cesión no entraña obligación ninguna para la Facultad de Ingeniería y Tecnología, que podrá o no ejercitar los derechos cedidos.

Se firma la presente en la ciudad de Montemorelos Nuevo León, a los 16 días del mes de abril de 2015.

Firma

CURP: zalb930802hcsvdn05

Observaciones (opcional)

CONTROL PROACTIVO DE TRÁFICO VEHICULAR MEDIANTE MACHINE LEARNING Y COMPUTACIÓN AUTÓNOMA

Informe de Investigación
presentado en cumplimiento parcial
de los requisitos para el grado
de Ingeniero en Sistemas
Computacionales

Por

Benjamín Zavala Ledesma

Aprobada por la comisión

Germán Harvey Alférez

Dr. Germán Harvey Alférez Salinas
Asesor Principal

Ing. Filiberto Grajeda Piedra
Evaluador

MC. Jair A. del Valle López
Evaluador

MC. Jair A. del Valle López
Coordinador de Pregrado

MC. Alejandro W. García Mendoza
Director de Facultad

DECLARACIÓN DE INTEGRACIÓN DE LA FE

‘‘Pero hágase todo decentemente y con orden."

1 Corintios 14:40

"Pues Dios no es Dios de confusión, sino de paz."
1 Corintios 14:33

Dios es un Dios de orden. Él se complace en un mundo donde todo tiene su lugar y donde todo tiene un propósito. En la confusión y en el desorden hay intranquilidad, estrés, y pesar. La fe se refleja en este trabajo al buscar un ambiente más ordenado y sin caos.

Tabla de Contenidos

Control proactivo del tráfico vehicular mediante Machine Learning y Computación Autónoma

Benjamín Zavala Ledesma
Global Software Lab
Facultad de Ingeniería y Tecnología, Universidad de Montemorelos
Montemorelos, N.L., México

***Abstract*—Con el crecimiento excesivo de las ciudades modernas se generan grandes problemas en la administración ciudadana. Uno de estos problemas es el control del flujo vehicular en horas pico. En este trabajo proponemos una solución al problema del control vehicular mediante una aproximación proactiva basada en Machine Learning. Mediante nuestra solución, un sistema de control de tráfico aprende acerca del flujo vehicular con el fin de prevenir futuros problemas de largas colas esperando en los semáforos. La arquitectura del sistema de tráfico se basa en los principios de Computación Autónoma con el fin de cambiar los temporizadores de los semáforos automáticamente. Una simulación de las vías en una ciudad inteligente y una herramienta basada en Weka fueron creadas para validar nuestra aproximación.**

Keywords: Machine Learning, Control Proactivo, Tráfico, Ciudades Inteligentes, Computación Autónoma.

I. INTRODUCCIÓN

A. Antecedentes y estado del arte

En los últimos años se ha venido dando un efecto de migración masiva del campo a la ciudad. Por ejemplo, en México hoy en día más del 70% de la población vive en la ciudad [1]. Hay un crecimiento exponencial en la población de las ciudades convirtiéndolas en mega ciudades. Este fenómeno genera muchos problemas, siendo uno de ellos las congestiones vehiculares [2]. Este problema afecta negativamente la calidad de vida de los ciudadanos al aumentar el tiempo de viaje, generando estrés y pérdidas económicas, y produciendo un incremento de la contaminación ambiental.

1) Estrategias computacionales de control de tráfico vehicular

Actualmente se han implementado estrategias de tiempo fijo y estrategias sensibles al tráfico para el manejo del flujo vehicular. Estas estrategias se presentan a continuación.

a) Estrategias de tiempo fijo

Las estrategias de tiempo fijo son ajustadas para periodos de tiempo largos donde dichos parámetros se asumen constantes. Lo anterior puede, entonces, resultar poco acertado en contextos con demandas de alta variabilidad o con usual presencia de condiciones no convencionales (e.g. accidentes, disturbios, o eventos inesperados).

Dentro de esta categoría se encuentra SIGSET, que en base a los patrones de flujo vehicular en un cruce calcula el tiempo del semáforo en cada ciclo. Este es un sistema muy conocido por los ingenieros de tráfico [3]. SIGSET trabaja de forma aislada en cada cruce y asigna tiempos fijos a los semáforos.

b) Estrategias sensibles al trafico

Las estrategias de control sensibles al tráfico ejecutan su lógica con base en mediciones de tráfico realizadas en tiempo real en las entradas a las intersecciones. Para llevar a cabo estas mediciones es necesario contar con algún tipo de detectores de tráfico.

Dentro de los métodos sensibles al tráfico hay dos métodos reactivos que solucionan problemas cuando estos ya son evidentes. Por un lado, hay aproximaciones que detectan la presencia de mucho tráfico en un cruce y modifican los tiempos de los semáforos para dar preferencia a la dirección con más tráfico. Por otra parte, otras soluciones son adaptativas. En estas soluciones se implementan redes de semáforos con planes de acción en conjunto para la optimización del flujo vehicular [4].

Otro tipo de estrategias sensibles al tráfico son las proactivas. Por ejemplo, RHODES[5] toma como datos de entrada la medición en tiempo real del flujo de tráfico, y controla el flujo vehicular a través de la red. El sistema utiliza una arquitectura de control que descompone el problema de control de tráfico en varias sub-problemas interconectados de forma jerárquica y predice los flujos de tráfico en los niveles adecuados de resolución. Este enfoque permite diferentes módulos de optimización para la solución de los sub-problemas jerárquicos, y utiliza una estructura de datos y un enfoque computador / comunicación que permiten la solución rápida de los sub-problemas. Este enfoque depende de un módulo de control central. Argumentamos que un enfoque más descentralizado podría ser utilizado para distribuir los cálculos en el lugar en donde pasa el tráfico. De esta manera, los costes y la complejidad relacionados con la infraestructura de comunicación podrían reducirse.

2) Conceptos Subyacentes

La solución que se propone en este trabajo es inteligente, autónoma, y proactiva. Estos conceptos subyacentes se describen a continuación.

a) Machine Learning

Machine Learning es un término usado para abarcar una amplia variedad de técnicas usadas para descubrir patrones y relaciones en grupos de datos. El objetivo primario de cualquier algoritmo de Machine Learning es descubrir relaciones significativas en sets de datos de entrenamiento y producir una generalización de estas relaciones que puedan ser usadas para interpretar nuevos datos desconocidos [6].

Dentro de los métodos de Machine Learning están las pronosticaciones. *Pronosticar* es el proceso de realizar afirmaciones sobre eventos cuyos resultados aún no han sido observados [7].

b) Computación Autónoma

Inspirada en la biología, la computación autónoma ha evolucionado como una disciplina para crear software que se auto-gestiona en un intento por superar las complejidades de mantener los sistemas de manera eficaz. La computación autónoma cubre un amplio espectro de la computación en dominios tan diversos como los dispositivos móviles y la automatización del hogar, demostrando su viabilidad y valor en la automatización de tareas tales como la instalación, la curación, y la actualización. Ya que realizar adaptaciones manuales es una tarea difícil (y a veces imposible), nuestra propuesta se basa en el modelo de referencia de IBM para los ciclos de control autónomos (que a veces se llama el bucle MAPE-K) [8].

c) Adaptaciones Proactivas

Por un lado, las adaptaciones reactivas se realizan en respuesta a un incidente. Por otro lado, las adaptaciones proactivas se llevan a cabo de antemano (es decir, antes de que un incidente afecte negativamente el sistema) [9]. Los mecanismos de adaptación reactivos pueden causar incrementos en el tiempo de ejecución y pérdida financiera, que pueden conducir a la insatisfacción del usuario [10]. Los enfoques proactivos tratan de resolver estos problemas mediante la detección de la necesidad de una adaptación antes de que el problema sea evidente.

B. Problema

En esta sección se presenta la declaración del problema y la justificación.

1) Declaración del problema

El común denominador de las soluciones actualmente implementadas para el control de tráfico en ciudades inteligentes consiste en que éstas esperan a que un evento suceda (e.g. una larga cola en un semáforo) para generar una solución a dicho evento. Estas soluciones se pueden denominar como reactivas. No obstante, creemos que para que las ciudades lleguen a ser verdaderamente inteligentes, es necesario contar con soluciones proactivas que anticipen los problemas de tráfico y eviten que estos problemas se hagan evidentes.

2) Justificación

En esta sección se presenta una simulación de las vías de una ciudad que justifica el problema del tráfico. Esta simulación consta de un cruce vehicular de dos vías unidireccionales. Con el fin de simplificar la simulación, los semáforos cambian únicamente entre luz verde y luz roja.

En una vía, la velocidad del flujo vehicular es determinada tanto como por la reglamentación vigente como por las intersecciones existentes. La congestión en algún semáforo se da cuando dicho semáforo permite el paso a una cantidad de vehículos menor a la cantidad de vehículos que llegan a la cola.

En la simulación se asignó a cada semáforo un valor de tiempo "*x*" para cada etapa (*rojo* ó *verde*). Además se determinó cuanto tiempo tardan en promedio los vehículos en cruzar la intersección ("*y*"). En base a estos datos se calculó cuántos carros pueden cruzar el semáforo cuando éste se encuentra en verde. Asimismo se asignó un valor aleatorio de flujo de entrada a la cola del semáforo "*i*".

Como se puede ver en la *Fig. 1*, la cantidad de vehículos tiende a incrementarse linealmente conforme pasa el tiempo. En nuestra simulación se generó esta tendencia cuando "*x*" toma un valor igual a 15 segundos, "*i*" tiene valores entre 0 y 9, y "*y*" es igual a 3 segundos.

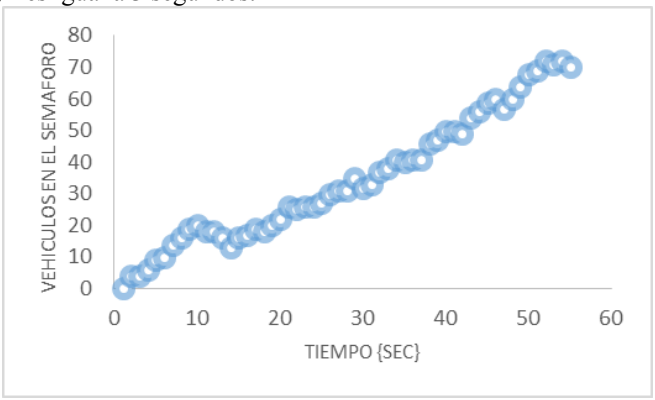

Fig. 1. Resultados de la simulación del tráfico en una ciudad inteligente

C. Objetivo

En el presente trabajo se presenta una solución proactiva para el control del tráfico vehicular mediante Machine Learning y Computación Autónoma. En primer lugar, nuestra solución analiza datos del nivel de tráfico mediante Machine Learning. Mediante esta técnica de inteligencia artificial, el sistema toma decisiones proactivas con respecto a datos históricos del tráfico. Con el fin de realizar ajustes autónomos en los semáforos, el sistema se auto-adapta mediante los principios de computación autónoma de IBM [11]. La eficiencia de nuestra solución se demuestra mediante una simulación del tráfico en una ciudad inteligente. Para predecir los problemas que pueden aparecer, la herramienta se basa en la API de Weka [12].

D. Preguntas e hipótesis

I. ¿Machine Learning puede servir para que se tomen decisiones proactivas con el fin de evitar problemas en el flujo vehicular?

II. ¿Un sistema basado en los principios de Computación Autónoma puede servir para que los temporizadores de los semáforos se cambien de forma automática?

III. ¿Es posible probar una aproximación de administración automática y proactiva de tráfico mediante una simulación computacional?

La hipótesis es la siguiente: "La utilización de Machine Learning y Computación Autóma puede resolver problemas en el control de tráfico vehicular de forma proactiva."

II. MÉTODO

La solución se basa en el ciclo MAPE-K (ver la *Fig. 2*). En el componente Monitor se realiza un periodo de entrenamiento en donde se recolectan los datos del tráfico mediante sensores. La tarea de observación del tráfico está a cargo del Observador de Tráfico. El Observador de Tráfico también está a cargo de detectar los momentos en donde ha habido alguna violación de algún Nivel de Acuerdo de Servicio (SLA en sus siglas en inglés) esperado de tráfico en cola. Luego, después de haber terminado el entrenamiento, el plug-in de predicciones de Weka analiza los datos recolectados en el componente de Análisis y predice posibles problemas de tráfico. A continuación, el Planeador de Adaptaciones en el componente de Planeación calcula los cambios necesarios en los temporizadores de los semáforos con el fin de prevenir problemas de tráfico de forma proactiva. Después de realizar la planeación, en el componente de Ejecución el Modificador de Temporizador realiza los cambios necesarios en los temporizadores del semáforo mediante actuadores. A continuación se describen nuestra solución en base a los componentes del ciclo MAPE-K. Nuestra solución se describe en base a la simulación de tráfico descrita anteriormente.

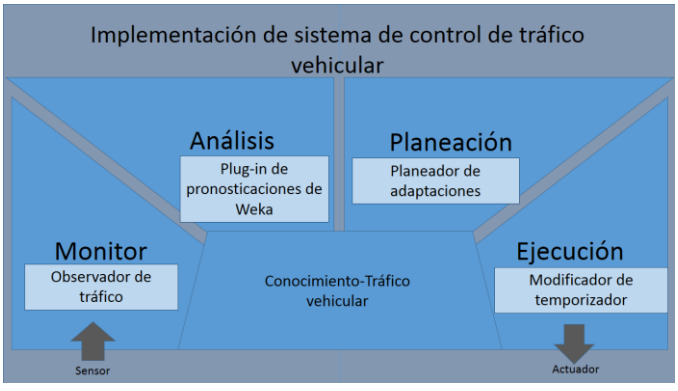

Fig.2 Ciclo MAPE-K de nuestra aproximación

A. Monitor

La supervisión implica capturar propiedades del entorno que son de importancia para las auto-propiedades del sistema. En nuestro caso, estamos interesados en la observación del tráfico. Para ello, proponemos el Observador de Tráfico, que es una herramienta que observa el tráfico a través de sensores. En nuestro caso, el Observador de Tráfico comprueba periódicamente la actividad en la simulación de tráfico.

Al iniciar la simulación se crean dos semáforos, y a cada uno se le asigna un estado *(Verde o Rojo)* y una cantidad aleatoria de vehículos. La etapa preventiva (amarillo) de un semáforo fue incluida dentro de la fase de paso (verde). Esto con la finalidad de simplificar la simulación.

El Observador de Trafico detecta si la cantidad de vehículos en cola en un determinado momento en el semáforo es mayor al SLA (i.e., violación del SLA). Si es así, este evento se guarda en un log con extensión .arff. Los archivos con extensión arff tienen el formato necesario para poder ejecutar pronosticaciones sobre estos datos según las librerías de Weka. Específicamente, el formato de archivo .arff requiere que los datos tengan un encabezado especial, tal como en el siguiente ejemplo:

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

En primer lugar existe una relación de datos (*@relation A*), y esta relación tiene dos atributos ambos de tipo numérico: segundos (*@attribute seconds numeric*) y carros (@attribute *cars numeric*). Después de esto se anotan los datos del evento problemático (i.e., una violación al SLA) (*@data*). En el ejemplo de arriba, existe una violación al SLA en el segundo 60 de ejecución de la simulación en donde hay 7 carros en cola, cuando el SLA indica que como sumo debería haber 3 carros en la cola.

B. Análisis

El objetivo de esta fase es detectar, de forma proactiva, los problemas que puedan ocurrir a futuro en el tráfico de acuerdo a los resultados capturados en el registro generado por el Observador de Trafico.

Para llevar a cabo la predicción de problemas de tráfico, se utilizó el plug-in de Pronósticos de Weka. Este plug-in puede cargar o importar un modelo de pronóstico de series de tiempo y utilizarlo para generar un pronóstico para pasos de tiempo futuros independientes de la final de los datos históricos de entrada [13]. De los métodos para realizar predicciones, en este trabajo se eligió el Perceptron Multicapa debido al menor porcentaje de error mostrado después de haber realizado pruebas con los distintos métodos presentes en Weka (Gaussian Process, Kernel Regression, Linear Regression, Multilayer Perceptron, y SMOreg).

Para que el Pronosticador funcione se le dan los siguientes parámetros sobre los cuales trabajar: tipo de dato a predecir, medida de tiempo de los datos de entrenamiento, y la cantidad de épocas a predecir. En nuestro caso, el Pronosticador predice la cantidad de carros que llegaran al semáforo después del entrenamiento siguiendo la medida de tiempo de los datos de

entrenamiento. Los datos que el Pronosticador genera son almacenados en un archivo de texto.

Por ejemplo, un posible entrenamiento (i.e., la observación del tráfico en la simulación durante cierto tiempo) concluyó en el segundo 1,800 de ejecución. Durante este tiempo se encontraron 25 violaciones al SLA. Con estos datos se realiza la predicción. Específicamente, el Pronosticador predice que si se sigue con el valor del temporizador actual de algún semáforo, entonces en el segundo 1,860 llegará un número mayor de carros que los especificados en el SLA. Mediante el pronóstico se logra predecir un problema que aún no ha sucedido con base a datos históricos.

C. Planeación

Es en esta fase se planea cómo solucionar de forma automática los problemas de tráfico predichos en la fase anterior luego de haber realizado el entrenamiento previo. Específicamente, en esta fase se calculan los tiempos que deberán tener los semáforos para evitar que los problemas predichos por la fase anterior del ciclo se materialicen.

Durante la planeación se lee el archivo creado en el componente de Análisis. Cada entrada en este archivo es un problema a solucionar (e.g. alguna violación de SLA). En esta fase se ha propuesto el Planeador de Adaptaciones, una herramienta que ejecuta los siguientes pasos para planear un cambio en el temporizador del semáforo:

1. El Planeador de Adaptaciones guarda en una variable el texto recuperado del archivo de la fase anterior. Por ejemplo, el Planeador de Adaptaciones toma el dato de 35 carros que violarán el SLA en algún tiempo futuro (después de haber terminado la ejecución del entrenamiento) de acuerdo a la predicción en la fase de Análisis.
2. El Planeador de Adaptaciones realiza la siguiente operación:

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

   "*d*" es el valor leído en el paso 1. "*tCross*" es el tiempo promedio que tarda un vehículo en pasar el cruce. "*timeR*" es el nuevo tiempo de semáforo. El resultado de la operación se transforma a tipo de dato entero.
3. Se guarda el valor de "*timeR*" que corresponde a la solución a un problema (i.e., alguna violación del SLA). Por ejemplo, al aplicar la fórmula anterior tenemos que si un vehículo tarda 3 segundos en pasar la intersección (*tCross*) y se esperan 35 carros que llegaran al semáforo (*d*) obtenemos que se necesitan 105 segundos (*timeR*) para que los 35 carros puedan pasar la intersección.

Cada uno de los valores calculados en esta fase corresponde a la solución de cada uno de los problemas previstos en la fase de Análisis.

D. Ejecución

El objetivo de esta fase consiste en realizar los cambios en los temporizadores de los semáforos de acuerdo a los resultados de la fase de Planeación. En esta fase se busca que los problemas predichos no se materialicen. Para realizar estos cambios se simuló el funcionamiento de actuadores cuyo objetivo es tomar el nuevo tiempo de semáforo, de los datos generados en la etapa anterior, y modificar el temporizador del semáforo.

Por ejemplo, en la fase de Planeación se obtuvo que se necesitan 105 segundos para que 35 carros puedan pasar la intersección. Por lo tanto, en esta fase nuestro Modificador de Temporizador le asigna un tiempo de 105 segundos al temporizador del semáforo. De esta manera cuando 35 carros llegan al semáforo este ya tiene preparado suficiente tiempo para permitir el paso de todos ellos.

III. RESULTADOS

En esta sección se describe el prototipo que se ha creado para demostrar nuestra aproximación. El prototipo fue creado en Java y la API de Weka que puede ser utilizada mediante Java [14]. El código fuente se presenta en el Apéndice al final de este documento.

La GUI del prototipo está dividida en tres áreas (ver la *Fig. 3*). En el área 1 se muestran las condiciones bajo las cuáles se realiza el control del tráfico. Específicamente se determinan el tiempo de los semáforos en rojo, el tiempo de ejecución de la simulación del tráfico, el tiempo del cruce, y el SLA esperado. En los campos de texto del área 2 se muestran los eventos en los que se violó el SLA en cada uno de los semáforos. La primera columna es el tiempo en segundos en el que sucedió el evento (i.e., la violación del SLA) y la segunda columna es la cantidad de carros con los cuáles se violó el SLA. En el área 3 se muestra cuantas veces se superó el SLA en cada semáforo antes y después de la implementación de Machine Learning.

El prototipo se entrena con los valores establecidos en la sección 1 de la *Fig 3*. En la ejecución se ve cómo durante la captura de datos para el entrenamiento la cantidad de veces que se violó el SLA es bastante alto en cada semáforo (e.g. 59 violaciones en el semáforo A y 57 en B). Luego, al implementar Machine Learning con pronósticos la cantidad de violaciones del SLA desciende drásticamente (17

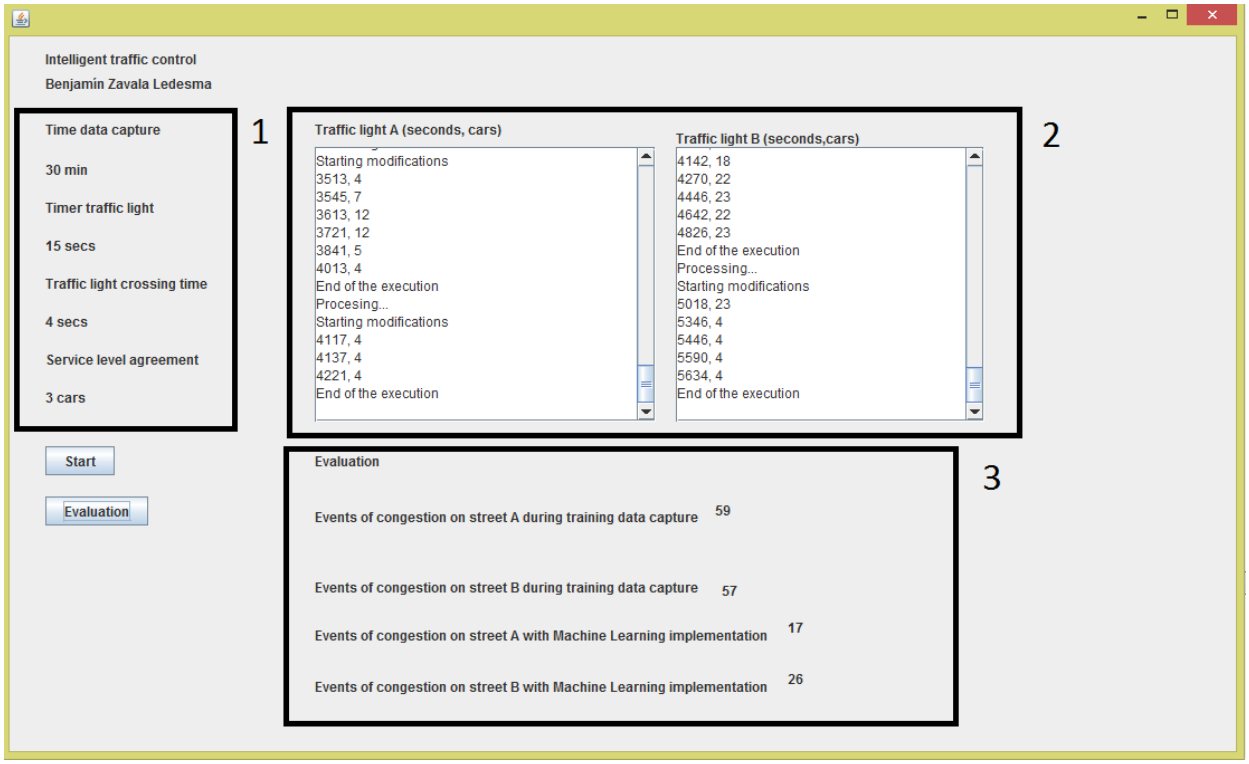

Fig.3 Ejecución del prototipo

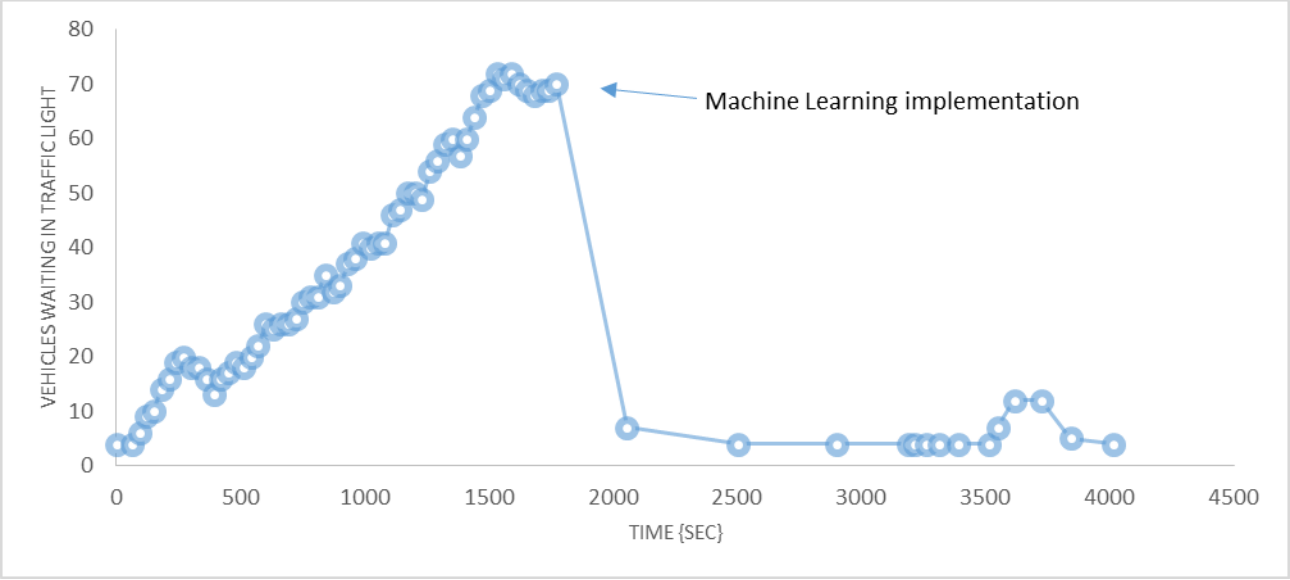

Fig.4 Ejecución completa de la herramienta

violaciones en A y 26 en B). Con esto se puede ver que las incidencias de violación del SLA no son completamente eliminadas pero sí que disminuyen en gran manera.

En la *Fig. 4* se puede ver como con la implementación de Machine Learning la tendencia de incremento de vehículos en espera en un semáforo desciende de forma abrupta. La implementación de Machine Learning ocurre en el segundo 1,800. A partir de este momento se puede apreciar cómo las incidencias en el SLA se mantienen en un nivel bastante bajo.

Las Soluciones proactivas al problema del tráfico pueden ayudar a mejorar la calidad de vida de los habitantes de las grandes ciudades.

IV. Conclusiones y trabajo futuro

A. Discusión de resultados

Se obtuvieron resultados bastante prometedores mediante la utilización de Machine Learning y Computación Autónoma a través del ciclo MAPE-K en una herramienta para el control de tráfico vehicular. Con el uso de estas aproximaciones se podría buscar aliviar la congestión vehicular en las ciudades de forma proactiva, es decir, aún antes que el problema sea evidente.

B. Conclusiones de la Investigación

En este trabajo se propuso una solución proactiva al control de tráfico mediante el uso de Machine Learning y los principios de la Computación Autónoma. Al ser nuestra solución completamente autónoma podemos minimizar el factor del error humano a la hora de controlar eventos en las vías. Los objetivos propuestos en esta investigación se han cubierto en su totalidad.

C. Reflexión

Nuestra solución presenta varios beneficios para el desarrollo de las mega-ciudades: 1) evitar pérdidas económicas al permitir que un cargamento o un trabajador llegue en el menor tiempo posible a su destino. En un atasco, los vehículos deben avanzar muy lento y estar frenando continuamente. Esto significa una aceleración en el cambio de piezas desgastadas y un mayor gasto de gasolina; 2) flujo vehicular más constante disminuyendo así el tiempo que un vehículo está en la vía; 3) aumentar el desempeño laboral del ciudadano mejorando la puntualidad en sus compromisos diarios al desplazarse de forma más rápida; 4) disminuir la emisión de agentes contaminantes del medio ambiente (CO_2) al mantener un vehículo menor tiempo encendido; y 5) disminuir la cantidad de accidentes vehiculares en horas pico. Estos accidentes pueden ser causados por el stress producido en los atascos [15].

D. Recomendaciones

La solución creada debe ser extendida para controlar más de un cruce. En el momento solo trabaja en un cruce. Además, con el fin de validar la solución es necesario contar con ambientes reales en donde probarla. Este aspecto llevará a buscar asociaciones con el gobierno para probar la aproximación en vías reales. Con el fin de extender esta investigación se buscará la creación de vínculos con proyectos europeos enfocados en ciudades inteligentes.

E. Futuros Aportes

Como trabajo futuro buscaremos implementar un módulo de visión por computador para un control en vivo del tráfico a través de las cámaras de tránsito. También se busca desarrollar una aplicación móvil a través de la cual el usuario pueda hacer consultas sobre el estado del tráfico y encontrar la ruta más óptima entre el punto de partida y el punto de destino obteniendo datos en tiempo real directamente de la central del sistema de control vehicular.

V. Referencias

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

[2] SIEMENS. (March 5th). *Siemens traffic solutions*. Available: https://www.swe.siemens.com/spain/web/es/industry/mobility/Documents/traffic.pdf

[3] R. E. Allsop, "Computer program SIGSET for calculating delay-minimissing traffic signal timings description and manual for users," 1981.

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

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

[6] S. Mitchell, "The application of machine learning techniques to time-series data," Citeseer, 1995.

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

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

[9] A. Metzger, "Towards accurate failure prediction for the proactive adaptation of service-oriented systems," presented at the Proceedings of the 8th workshop on Assurances for self-adaptive systems, Szeged, Hungary, 2011.

[10] R. R. Aschoff and A. Zisman, "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, 2012.

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

[12] Pentaho. (March 11th). *Time Series Analysis and Forecasting with Weka*. Available: http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka

[13] Pentaho. (2013, March 18th). *Using the Weka Forecasting Plugin*. Available: http://wiki.pentaho.com/display/DATAMINING/Using+the+Weka+Forecasting+Plugin

[14] Pentaho. (2013, March 24th). *Weka 3: Data Mining Software in Java*. Available: http://www.cs.waikato.ac.nz/ml/weka/

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

VI. Apéndice – Código Fuente

```
[1]   package tesis;
[2]
[3]   import java.awt.Desktop;
[4]   import java.awt.FlowLayout;
[5]   import java.awt.List;
[6]   import java.awt.event.ActionEvent;
[7]   import java.awt.event.ActionListener;
[8]   import java.io.BufferedReader;
[9]   import java.io.BufferedWriter;
[10] import java.io.File;
[11] import java.io.FileInputStream;
[12] import java.io.FileNotFoundException;
[13] import java.io.FileOutputStream;
[14] import java.io.FileReader;
[15] import java.io.FileWriter;
[16] import java.io.IOException;
[17] import java.io.InputStream;
[18] import java.io.LineNumberReader;
[19] import java.io.OutputStream;
[20] import java.io.PrintWriter;
[21] import java.lang.Thread.State;
[22] import java.text.SimpleDateFormat;
[23] import java.util.ArrayList;
[24] import java.util.Date;
[25] import java.util.Random;
[26] import java.util.Scanner;
[27] import java.util.logging.Level;
[28] import java.util.logging.Logger;
[29] import javax.swing.JFrame;
[30] import javax.swing.JLabel;
[31] import javax.swing.ToolTipManager;
[32] import weka.classifiers.evaluation.NumericPrediction;
[33] import weka.classifiers.functions.MultilayerPerceptron;
[34] import weka.classifiers.timeseries.WekaForecaster;
[35] import weka.core.Instances;
[36]
[37] /**
[38]  *
[39]  * @author BzlZavLed
[40]  */
[41] public class teisiVista extends javax.swing.JFrame {
[42]
[43]     private volatile boolean term = true;
[44]     int red, date = 0, tSim;
[45]     Random ra1 = new Random();
[46]     Random ra2 = new Random();
[47]     Random ra3 = new Random();
[48]     Random ra4 = new Random();
[49]     ArrayList list = new ArrayList();
[50]     ArrayList list2 = new ArrayList();
[51]     ArrayList semA = new ArrayList();
[52]     ArrayList semB = new ArrayList();
[53]     ArrayList predA = new ArrayList();
[54]     ArrayList predB = new ArrayList();
[55]      ArrayList incidenciasA = new ArrayList();
[56]       ArrayList incidenciasB = new ArrayList();
[57]     boolean frep = true;
[58]     boolean ea1 = false;  //rojo
[59]     boolean ea2 = true;   // verde
[60]     boolean traf = false;
[61]     boolean state = true;
[62]     boolean training = false;
[63]     boolean tiempo = false;
[64]     boolean tiempo2 = false;
[65]     int contadorA1 = 0;
[66]     int contadorA2 = 0;
[67]     int levelAgreement = 0;
[68]     int level;
[69]     int repeticiones;
[70]     boolean slaExc = false;
[71]     String sFicheroA = "A.arff";
[72]     String sFicheroB = "B.arff";
[73]     String sFicheroA1 = "A1.arff";
[74]     String sFicheroB1 = "B1.arff";
[75]     String sPredA = "predA.txt";
[76]     String sPredB = "predB.txt";
[77]     String respaldoAB = "respaldogenA.csv";
[78]     String respaldoAB1 = "respaldogenB.csv";
[79]     String respaldoAeje = "respaldogenAeje.csv";
[80]     String respaldoBeje = "respaldogenBeje.csv";
[81]     String sPredResp = "predRespA.txt";
[82]     String sPredRespB = "predRespB.txt";
[83]     String modResp = "modResp.txt";
[84]     String str;
[85]     String strB;
[86]     String h ;
[87]
[88]     int dif1 = 0;
[89]     int dif2 = 0;
[90]     int tCruce = 0;
[91]     int timer = 0;
[92]     int timerB = 0;
[93]     int cont = 1;
[94]     int rep = 0;
[95]     int tRojoOri;
[96]     int tRojoA;
[97]     int tRojoB;
[98]     int carros = 0;
[99]     int carrosB = 0;
[100]         int carrosA = 0;
```

```
[101]     int lista = 0;
[102]     int listaCont = 0;
[103]     int listaB = 0;
[104]     int listaContB = 0;
[105]     int contIncA = 0;
[106]     int contIncB = 0;
[107]     int contIncAT = 0;
[108]     int contIncBT = 0;
[109]     int contEntA = 0;
[110]     int contEntB = 0;
[111]     int contciclos = 0;
[112]     int val = 0;
[113]     int val2 = 0;
[114]
[115]     int contMa = 0;
[116]     File fichero;
[117]     File ficheroB;
[118]     File ficheroA1;
[119]     File ficheroB1;
[120]     File fichero1;
[121]     File fichero2;
[122]     File ficheroPredA;
[123]     File ficheroPredB;
[124]     File respGeneA;
[125]     File respGeneB;
[126]     File respGeneAeje;
[127]     File respGeneBeje;
[128]     BufferedWriter respaldogeneralA;
[129]     BufferedWriter respaldogeneralB;
[130]     BufferedWriter respaldogeneralAeje;
[131]     BufferedWriter respaldogeneralBeje;
[132]     BufferedWriter respaldo;
[133]     BufferedWriter bw;
[134]     BufferedWriter bwB;
[135]     BufferedWriter bwA1;
[136]     BufferedWriter bwB1;
[137]     BufferedWriter bwModA;
[138]     BufferedWriter bwModB;
[139]     BufferedWriter bwPredA;
[140]     BufferedWriter bwPredB;
[141]     BufferedWriter resp;
[142]
[143]     BufferedReader lec = null;
[144]     BufferedReader lecB = null;
[145]     Thread simulation;
[146]     File ficheroPredResp = new File(sPredResp);
[147]     BufferedWriter bwPredResp = null;
[148]     File ficheroPredRespB = new File(sPredRespB);
[149]     BufferedWriter bwPredRespB = null;
[150]     File respaldoGen = new File(respaldoAB);
[151]     BufferedWriter respGen = null;
[152]
[153]
[154]     /**
[155]      * Creates new form teisiVista
[156]      */
[157]     public teisiVista() {
[158]
[159]         initComponents();
[160]         tsimular.setToolTipText("La cantidad de tiempo a simular para el entrenamiento, debe ser en segundos, por ejemplo 1 dia = 86400 segundos");
[161]
[162]         trojosim.setToolTipText("Durante la simulacion (entrenamiento), los semaforos tienen tiempos estaticos en rojo y verde (amarillo no incluido), \n"
[163]                 + "el valor ingresado aqui sera el tiempo que tarde el semaforo en un color, debe ser en segundos ");
[164]         tcrucecar.setToolTipText("El valor ingresado aqui es la cantidad de tiempo que tarda un vehiculo en pasar la linea del semaforo correpondiente,\n"
[165]                 + " este valor tambien es en segundos, para mayor exactitud se recomiendan valores entre 2-5 segundos");
[166]     }
[167]
[168]     /**
[169]      * This method is called from within the constructor to initialize the form.
[170]      * WARNING: Do NOT modify this code. The content of this method is always
[171]      * regenerated by the Form Editor.
[172]      */
[173]     @SuppressWarnings("unchecked")
[174]     // <editor-fold defaultstate="collapsed" desc="Generated Code">
[175]     private void initComponents() {
[176]
[177]         jLabel1 = new javax.swing.JLabel();
[178]         jLabel2 = new javax.swing.JLabel();
[179]         tsimular = new javax.swing.JLabel();
[180]         trojosim = new javax.swing.JLabel();
[181]         tcrucecar = new javax.swing.JLabel();
[182]         Inicio = new javax.swing.JButton();
[183]         jLabel8 = new javax.swing.JLabel();
[184]         jLabel9 = new javax.swing.JLabel();
[185]         jScrollPane1 = new javax.swing.JScrollPane();
[186]         a = new javax.swing.JTextArea();
[187]         jScrollPane2 = new javax.swing.JScrollPane();
[188]         b = new javax.swing.JTextArea();
[189]         evaluacion = new javax.swing.JButton();
[190]         jLabel7 = new javax.swing.JLabel();
[191]         jLabel20 = new javax.swing.JLabel();
[192]         jLabel21 = new javax.swing.JLabel();
[193]         incB = new javax.swing.JLabel();
[194]         incA = new javax.swing.JLabel();
[195]         jLabel24 = new javax.swing.JLabel();
```

```
[196]           jLabel25 = new javax.swing.JLabel();
[197]           incEjeA = new javax.swing.JLabel();
[198]           incEjeB = new javax.swing.JLabel();
[199]           jLabel3 = new javax.swing.JLabel();
[200]           jLabel4 = new javax.swing.JLabel();
[201]           jLabel5 = new javax.swing.JLabel();
[202]           jLabel6 = new javax.swing.JLabel();
[203]           jLabel10 = new javax.swing.JLabel();
[204]
[205]       setDefaultCloseOperation(javax.swing.WindowConstants.EXIT
            _ON_CLOSE);
[206]
[207]           jLabel1.setText("Intelligent traffic control");
[208]
[209]           jLabel2.setText("Benjamín Zavala Ledesma");
[210]
[211]           tsimular.setText("Time data capture");
[212]
[213]           trojosim.setText("Timer traffic light");
[214]
[215]           tcrucecar.setText("Traffic light crossing time");
[216]
[217]           Inicio.setText("Start");
[218]           Inicio.addActionListener(new
            java.awt.event.ActionListener() {
[219]               public                                  void
            actionPerformed(java.awt.event.ActionEvent evt) {
[220]                   InicioActionPerformed(evt);
[221]               }
[222]           });
[223]
[224]           jLabel8.setText("Traffic light A (seconds, cars)");
[225]
[226]           jLabel9.setText("Traffic light B (seconds,cars)");
[227]
[228]           a.setColumns(20);
[229]           a.setRows(5);
[230]           jScrollPane1.setViewportView(a);
[231]           a.setEditable(false);
[232]           Console.useJTextComponent(a);
[233]
[234]           b.setColumns(20);
[235]           b.setRows(5);
[236]           jScrollPane2.setViewportView(b);
[237]           b.setEditable(false);
[238]           Console.useJTextComponent2(b);
[239]
[240]           evaluacion.setText("Evaluation");
[241]           evaluacion.addActionListener(new
            java.awt.event.ActionListener() {
[242]               public                                  void
            actionPerformed(java.awt.event.ActionEvent evt) {
[243]                   evaluacionActionPerformed(evt);
[244]               }
[245]           });
[246]
[247]           jLabel7.setText("Evaluation");
[248]
[249]           jLabel20.setText("Events of congestion on street A
            during training data capture ");
[250]
[251]           jLabel21.setText("Events of congestion on street B
            during training data capture ");
[252]
[253]           jLabel24.setText("Events of congestion on street A
            with Machine Learning implementation");
[254]
[255]           jLabel25.setText("Events of congestion on street B
            with Machine Learning implementation");
[256]
[257]           jLabel3.setText("Service level agreement");
[258]
[259]           jLabel4.setText("30 min");
[260]
[261]           jLabel5.setText("15 secs");
[262]
[263]           jLabel6.setText("4 secs");
[264]
[265]           jLabel10.setText("3 cars");
[266]
[267]           javax.swing.GroupLayout      layout      =      new
            javax.swing.GroupLayout(getContentPane());
[268]           getContentPane().setLayout(layout);
[269]           layout.setHorizontalGroup(
[270]
            layout.createParallelGroup(javax.swing.GroupLayout.Alignmen
            t.LEADING)
[271]               .addGroup(layout.createSequentialGroup()
[272]                   .addGap(32, 32, 32)
[273]
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
            t.Alignment.LEADING)
[274]                       .addComponent(jLabel1)
[275]                       .addComponent(tsimular)
[276]                       .addComponent(jLabel2)
[277]                       .addGroup(layout.createSequentialGroup()
[278]
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
            t.Alignment.LEADING)
[279]                           .addComponent(evaluacion)
[280]                           .addComponent(Inicio)
[281]                           .addComponent(trojosim)
[282]                           .addComponent(tcrucecar)
[283]                           .addComponent(jLabel3)
[284]                           .addComponent(jLabel4)
[285]                           .addComponent(jLabel5)
```

```
[286]                         .addComponent(jLabel6)
[287]                         .addComponent(jLabel10))
[288]                     .addGap(96, 96, 96)
[289]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[290]
    .addGroup(layout.createSequentialGroup()
[291]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[292]                         .addComponent(jScrollPane1,
    javax.swing.GroupLayout.PREFERRED_SIZE,          305,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[293]                         .addComponent(jLabel8))
[294]                     .addGap(18, 18, 18)
[295]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[296]                         .addComponent(jScrollPane2,
    javax.swing.GroupLayout.PREFERRED_SIZE,          276,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[297]                         .addComponent(jLabel9)))
[298]
    .addGroup(layout.createSequentialGroup()
[299]                         .addComponent(jLabel20)
[300]
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
[301]                         .addComponent(incA,
    javax.swing.GroupLayout.PREFERRED_SIZE,          79,
    javax.swing.GroupLayout.PREFERRED_SIZE))
[302]                     .addComponent(jLabel7)
[303]
    .addGroup(layout.createSequentialGroup()
[304]                         .addComponent(jLabel21)
[305]                         .addGap(18, 18, 18)
[306]                         .addComponent(incB,
    javax.swing.GroupLayout.PREFERRED_SIZE,          85,
    javax.swing.GroupLayout.PREFERRED_SIZE))
[307]
    .addGroup(layout.createSequentialGroup()
[308]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[309]                         .addComponent(jLabel24)
[310]                         .addComponent(jLabel25))
[311]                     .addGap(18, 18, 18)
[312]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[313]                         .addComponent(incEjeA,
    javax.swing.GroupLayout.PREFERRED_SIZE,          112,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[314]                         .addComponent(incEjeB,
    javax.swing.GroupLayout.PREFERRED_SIZE,          112,
    javax.swing.GroupLayout.PREFERRED_SIZE))))))
[315]                 .addContainerGap(225, Short.MAX_VALUE))
[316]         );
[317]         layout.setVerticalGroup(
[318]
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[319]             .addGroup(layout.createSequentialGroup()
[320]                 .addContainerGap()
[321]                 .addComponent(jLabel1)
[322]
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
[323]                 .addComponent(jLabel2)
[324]                 .addGap(18, 18, 18)
[325]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
[326]                     .addComponent(jLabel9)
[327]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
[328]                     .addComponent(tsimular,
    javax.swing.GroupLayout.PREFERRED_SIZE,          31,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[329]                     .addComponent(jLabel8,
    javax.swing.GroupLayout.PREFERRED_SIZE,          31,
    javax.swing.GroupLayout.PREFERRED_SIZE)))
[330]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
[331]                     .addGroup(layout.createSequentialGroup()
[332]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[333]
    .addGroup(layout.createSequentialGroup()
[334]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
[335]                         .addComponent(jScrollPane2,
    javax.swing.GroupLayout.PREFERRED_SIZE,          246,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[336]                         .addComponent(jScrollPane1,
    javax.swing.GroupLayout.PREFERRED_SIZE,          246,
    javax.swing.GroupLayout.PREFERRED_SIZE))
[337]                     .addGap(22, 22, 22)
[338]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
[339]                         .addComponent(jLabel7)
[340]                         .addComponent(Inicio)))
[341]
    .addGroup(layout.createSequentialGroup()
[342]                     .addGap(12, 12, 12)
[343]                     .addComponent(jLabel4)
[344]                     .addGap(18, 18, 18)
[345]                     .addComponent(trojosim)
```

```
[346]                         .addGap(18, 18, 18)
[347]                         .addComponent(jLabel5)
[348]                         .addGap(18, 18, 18)
[349]                         .addComponent(tcrucecar)
[350]                         .addGap(18, 18, 18)
[351]                         .addComponent(jLabel6)
[352]                         .addGap(18, 18, 18)
[353]                         .addComponent(jLabel3)
[354]                         .addGap(18, 18, 18)
[355]                         .addComponent(jLabel10)))
[356]                     .addGap(18, 18, 18)
[357]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
    t.Alignment.LEADING)
[358]                         .addComponent(jLabel20,
    javax.swing.GroupLayout.Alignment.TRAILING)
[359]                         .addComponent(incA,
    javax.swing.GroupLayout.Alignment.TRAILING,
    javax.swing.GroupLayout.PREFERRED_SIZE,          27,
    javax.swing.GroupLayout.PREFERRED_SIZE)))
[360]                     .addGroup(layout.createSequentialGroup()
[361]
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
    nt.RELATED)
[362]                         .addComponent(evaluacion)))
[363]                 .addGap(47, 47, 47)
[364]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
    t.Alignment.LEADING)
[365]                     .addComponent(incB,
    javax.swing.GroupLayout.PREFERRED_SIZE,          23,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[366]                     .addComponent(jLabel21))
[367]
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
    nt.RELATED,      javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE)
[368]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
    t.Alignment.TRAILING)
[369]                     .addComponent(incEjeA,
    javax.swing.GroupLayout.PREFERRED_SIZE,          30,
    javax.swing.GroupLayout.PREFERRED_SIZE)
[370]                     .addComponent(jLabel24))
[371]                 .addGap(17, 17, 17)
[372]
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
    t.Alignment.TRAILING)
[373]                     .addComponent(jLabel25)
[374]                     .addComponent(incEjeB,
    javax.swing.GroupLayout.PREFERRED_SIZE,          29,
    javax.swing.GroupLayout.PREFERRED_SIZE))
[375]                 .addGap(50, 50, 50))
[376]             );
[377]
[378]             pack();
[379]         }// </editor-fold>
[380]
[381]         private                                          void
    InicioActionPerformed(java.awt.event.ActionEvent evt) {
[382]             simulation = new Thread(new Sim());
[383]             simulation.start();
[384]
[385]
[386]         }
[387]
[388]         private                                          void
    evaluacionActionPerformed(java.awt.event.ActionEvent evt) {
[389]
[390]             int reps = tSim / red;
[391]             int cotaEntA = (contEntA*100)/(reps/2);
[392]             int cotaEntB = (contEntB*100)/(reps/2);
[393]             int cotaEjA = (contIncA*100)/(reps/2);
[394]             int cotaEjB = (contIncB*100)/(reps/2);
[395]
[396]             incA.setText(String.valueOf(contEntA));
[397]             incB.setText(String.valueOf(contEntB));
[398]
[399]             incEjeA.setText(String.valueOf(contIncA));
[400]             incEjeB.setText(String.valueOf(contIncB));
[401]         }
[402]
[403]         public static void main(String args[]) {
[404]
[405]             java.awt.EventQueue.invokeLater(new Runnable() {
[406]                 public void run() {
[407]                     new teisiVista().setVisible(true);
[408]
[409]                 }
[410]             });
[411]         }
[412]
[413]         // Variables declaration - do not modify
[414]         private javax.swing.JButton Inicio;
[415]         private javax.swing.JTextArea a;
[416]         private javax.swing.JTextArea b;
[417]         private javax.swing.JButton evaluacion;
[418]         private javax.swing.JLabel incA;
[419]         private javax.swing.JLabel incB;
[420]         private javax.swing.JLabel incEjeA;
[421]         private javax.swing.JLabel incEjeB;
[422]         private javax.swing.JLabel jLabel1;
[423]         private javax.swing.JLabel jLabel10;
[424]         private javax.swing.JLabel jLabel2;
[425]         private javax.swing.JLabel jLabel20;
[426]         private javax.swing.JLabel jLabel21;
[427]         private javax.swing.JLabel jLabel24;
[428]         private javax.swing.JLabel jLabel25;
```

```
[429]        private javax.swing.JLabel jLabel3;
[430]        private javax.swing.JLabel jLabel4;
[431]        private javax.swing.JLabel jLabel5;
[432]        private javax.swing.JLabel jLabel6;
[433]        private javax.swing.JLabel jLabel7;
[434]        private javax.swing.JLabel jLabel8;
[435]        private javax.swing.JLabel jLabel9;
[436]        private javax.swing.JScrollPane jScrollPane1;
[437]        private javax.swing.JScrollPane jScrollPane2;
[438]        private javax.swing.JLabel tcrucecar;
[439]        private javax.swing.JLabel trojosim;
[440]        private javax.swing.JLabel tsimular;
[441]        // End of variables declaration
[442]
[443]        public class Sim implements Runnable {
[444]
[445]            public void run() {
[446]                try {
[447]
[448]                    //writers
[449]                    respGeneA = new File(respaldoAB);
[450]                    respaldogeneralA = new BufferedWriter(new
     FileWriter(respaldoAB, true));
[451]                    respGeneB = new File(respaldoAB1);
[452]                    respaldogeneralB = new BufferedWriter(new
     FileWriter(respaldoAB1, true));
[453]                    respGeneAeje = new File(respaldoAeje);
[454]                    respaldogeneralAeje = new BufferedWriter(new
     FileWriter(respaldoAeje, true));
[455]                    respGeneBeje = new File(respaldoBeje);
[456]                    respaldogeneralBeje = new BufferedWriter(new
     FileWriter(respaldoBeje, true));
[457]
[458]                    fichero = new File(sFicheroA);
[459]                    bw = new BufferedWriter(new
     FileWriter(sFicheroA, true));
[460]                    ficheroB = new File(sFicheroB);
[461]                    bwB = new BufferedWriter(new
     FileWriter(sFicheroB, true));
[462]                    ficheroA1 = new File(sFicheroA1);
[463]                    ficheroB1 = new File(sFicheroB1);
[464]
[465]                    if(ficheroA1.exists() && ficheroB1.exists()){
[466]
[467]                        InputStream x = null;
[468]                        OutputStream y = null;
[469]                        InputStream i = null;
[470]                        OutputStream j = null;
[471]
[472]                        try {
[473]
[474]                            File file1 = new File(sFicheroA1);
[475]                            File file2 = new File(sFicheroA);
[476]                            File fileB = new File(sFicheroB1);
[477]                            File fileB1 = new File(sFicheroB);
[478]
[479]                            x = new FileInputStream(file1);
[480]                            y = new FileOutputStream(file2);
[481]                            i = new FileInputStream(fileB);
[482]                            j = new FileOutputStream(fileB1);
[483]
[484]                            byte[] buffer = new byte[1024];
[485]                            byte[] bufferB1 = new byte[1024];
[486]
[487]                            int length;
[488]                            int lengthB1;
[489]                            while ((length = x.read(buffer)) > 0 &&
     (lengthB1 = i.read(bufferB1)) > 0) {
[490]                                y.write(buffer, 0, length);
[491]                                j.write(bufferB1, 0, lengthB1);
[492]                            }
[493]
[494]                            if (x != null) {
[495]                                x.close();
[496]                            }
[497]                            if (y != null) {
[498]                                y.close();
[499]                            }
[500]                            if (i != null) {
[501]                                i.close();
[502]                            }
[503]                            if (j != null) {
[504]                                j.close();
[505]                            }
[506]
[507]                            System.out.println("files copied to
     originals in creation");
[508]                        } catch (IOException e) {
[509]                        }
[510]
[511]                    }else{
[512]
[513]                    bwA1 = new BufferedWriter(new
     FileWriter(sFicheroA1, true));
[514]
[515]                    bwB1 = new BufferedWriter(new
     FileWriter(sFicheroB1, true));
[516]                    bw.write("@relation A \n");
[517]                    bw.write("@attribute seconds numeric \n");
[518]                    bw.write("@attribute carros numeric @data \n");
[519]                    bw.write("@data \n");
[520]                    bw.write("\n");
[521]                    bwB.write("@relation B \n");
[522]                    bwB.write("@attribute seconds numeric \n");
[523]                    bwB.write("@attribute carros numeric @data
     \n");
```

```
[524]                 bwB.write("@data \n");
[525]                 bwB.write("\n");
[526]                 respaldogeneralA.write("segundos, carros \n");
[527]                 respaldogeneralB.write("segundos, carros \n");
[528]                 respaldogeneralAeje.write("segundos, carros \n");
[529]                 respaldogeneralBeje.write("segundos, carros \n");
[530]                 }
[531]
[532]                 File respaldo = new File(modResp);
[533]                 resp = new BufferedWriter(new
      FileWriter(respaldo, true));
[534]                 bwPredResp = new BufferedWriter(new
      FileWriter(sPredResp, true));
[535]
[536]
[537]                 Console.println("seconds,cars");
[538]                 Console.printline("seconds,cars");
[539]       //valores para simulacion
[540]
[541]                    tSim =  1800;
[542]
[543]                    levelAgreement = 3;
[544]
[545]                 /* if("30 segundos".equals(j)){
[546]                    red = 30;
[547]                    tRojoOri = 30;
[548]                 }
[549]                 if("1 minuto".equals(j)){
[550]                    red = 60;
[551]                    tRojoOri = 60;
[552]                 }*/
[553]
[554]                    red = 15;
[555]                    tRojoOri = 15;
[556]
[557]
[558]
[559]
[560]                 //if("2 segundos".equals(h)){
[561]                    //tCruce = 2;
[562]                 //}
[563]
[564]                    tCruce = 4;
[565]
[566]                 System.out.println("El tiempo de simulacion es "
      + tSim);
[567]                 //tCruce                                    =
      Integer.parseInt(carrosCruce.getText());
[568]                 System.out.println("El tiempo de cruce es " +
      tCruce);
[569]                 //levelAgreement                            =
      Integer.parseInt(SLA.getText());
[570]                 System.out.println("El nivel de acuerdo es " +
      levelAgreement);
[571]                 //red = Integer.parseInt(tRed.getText());
[572]                 System.out.println("El tiempo en rojo es " + red);
[573]
[574]                 int autos = red / tCruce;
[575]                 repeticiones = tSim / tRojoOri;
[576]
[577]
[578]                 System.out.println("La cantidad de ciclos de
      entrenamiento " + repeticiones);
[579]                 rep = tSim / red;
[580]
[581]                 carros = tRojoOri / tCruce;
[582]                 //int m = Integer.parseInt(predict.getText());
[583]       //inicio de ciclo
[584]                 //while (!Thread.currentThread().isInterrupted())
[585]                 while (contciclos!=rep){
[586]                    Thread prediction = new Thread(new predA());
[587]                    Thread prediction2 = new Thread(new
      predB());
[588]                    Thread mon = new Thread(new monitor());
[589]
[590]                    //pausa la ejecucion del hilo para esperar la
      respuesta de forecasting
[591]                    if (cont == repeticiones) {
[592]                       Console.println("Procesing...");
[593]                       Console.printline("Processing...");
[594]                       training = true;
[595]                       contIncAT+=contIncA;
[596]                       contIncBT+=contIncB;
[597]                       System.out.println("contTotalA                eje
      "+contIncAT);
[598]                       System.out.println("contTotalB                eje
      "+contIncBT);
[599]                       incidenciasA.add(contIncA);
[600]                       incidenciasB.add(contIncB);
[601]                       System.out.println("conteo de incidencias en
      A "+ contIncAT);
[602]                       System.out.println("conteo de incidencias en
      B "+ contIncBT);
[603]                       //contIncA =0;
[604]                       //contIncB = 0;
[605]
[606]                       bw.close();
[607]                       bwB.close();
[608]                       list.clear();
[609]                       list2.clear();
[610]                       InputStream inStream = null;
[611]                       OutputStream outStream = null;
[612]                       InputStream inStreamB = null;
[613]                       OutputStream outStreamB1 = null;
[614]
```

```
[615]                 try {
[616]
[617]                   File file1 = new File(sFicheroA);
[618]                   File file2 = new File(sFicheroA1);
[619]                   File fileB = new File(sFicheroB);
[620]                   File fileB1 = new File(sFicheroB1);
[621]
[622]                   inStream = new FileInputStream(file1);
[623]                   outStream = new FileOutputStream(file2);
[624]                   inStreamB = new FileInputStream(fileB);
[625]                   outStreamB1 = new FileOutputStream(fileB1);
[626]
[627]                   byte[] buffer = new byte[1024];
[628]                   byte[] bufferB1 = new byte[1024];
[629]
[630]                   int length;
[631]                   int lengthB1;
[632]                   while ((length = inStream.read(buffer)) > 0 && (lengthB1 = inStreamB.read(bufferB1)) > 0) {
[633]                     outStream.write(buffer, 0, length);
[634]                     outStreamB1.write(bufferB1, 0, lengthB1);
[635]                   }
[636]
[637]                   if (inStream != null) {
[638]                     inStream.close();
[639]                   }
[640]                   if (outStream != null) {
[641]                     outStream.close();
[642]                   }
[643]                   if (inStreamB != null) {
[644]                     inStreamB.close();
[645]                   }
[646]                   if (outStreamB1 != null) {
[647]                     outStreamB1.close();
[648]                   }
[649]
[650]                   System.out.println("Files Copied..");
[651]                 } catch (IOException e) {
[652]                 }
[653]
[654]                 prediction.start();
[655]                 prediction.join();
[656]                 prediction2.start();
[657]                 prediction2.join();
[658]                 mon.start();
[659]                 mon.join();
[660]
[661]                 fichero.delete();
[662]                 System.out.println("A.arff borrado en ciclo " + cont);
[663]                 ficheroB.delete();
[664]                 System.out.println("B.arff borrado en ciclo " + cont);
[665]
[666]                 System.out.println("Creando archivos para siguiente ciclo");
[667]                 fichero = new File(sFicheroA);
[668]                 bw = new BufferedWriter(new FileWriter(sFicheroA, true));
[669]                 System.out.println("A.arff creado en ciclo " + cont);
[670]                 ficheroB = new File(sFicheroB);
[671]                 bwB = new BufferedWriter(new FileWriter(sFicheroB, true));
[672]                 System.out.println("B.arff creado en ciclo " + cont);
[673]
[674]                 InputStream a = null;
[675]                 OutputStream a1 = null;
[676]                 InputStream b = null;
[677]                 OutputStream b1 = null;
[678]
[679]                 try {
[680]
[681]                   File file1 = new File(sFicheroA1);
[682]                   File file2 = new File(sFicheroA);
[683]                   File fileB = new File(sFicheroB1);
[684]                   File fileB1 = new File(sFicheroB);
[685]
[686]                   a = new FileInputStream(file1);
[687]                   a1 = new FileOutputStream(file2);
[688]                   b = new FileInputStream(fileB);
[689]                   b1 = new FileOutputStream(fileB1);
[690]
[691]                   byte[] buffer = new byte[1024];
[692]                   byte[] bufferB1 = new byte[1024];
[693]
[694]                   int length;
[695]                   int lengthB1;
[696]                   while ((length = a.read(buffer)) > 0 && (lengthB1 = b.read(bufferB1)) > 0) {
[697]                     a1.write(buffer, 0, length);
[698]                     b1.write(bufferB1, 0, lengthB1);
[699]                   }
[700]
[701]                   if (a != null) {
[702]                     a.close();
[703]                   }
[704]                   if (a1 != null) {
[705]                     a1.close();
[706]                   }
[707]                   if (b != null) {
[708]                     b.close();
```

```
[709]                     }
[710]                     if (b1 != null) {
[711]                        b1.close();
[712]                     }
[713]
[714]                     System.out.println("Files Copied Back..");
[715]                  } catch (IOException e) {
[716]                  }
[717]
[718]                  System.out.println("Cargando nuevos tiempos de semaforo");
[719]                  lec = new BufferedReader(new FileReader("modA.txt"));
[720]                  lecB = new BufferedReader(new FileReader("modB.txt"));
[721]
[722]                  while ((str = lec.readLine()) != null) {
[723]                     list.add(str);
[724]                     lista = list.size();
[725]                  }
[726]                  while ((strB = lecB.readLine()) != null) {
[727]                     list2.add(strB);
[728]                     listaB = list2.size();
[729]                  }
[730]                  lec.close();
[731]                  lecB.close();
[732]                  fichero1.delete();
[733]                  fichero2.delete();
[734]                  if (fichero1.exists() && fichero2.exists()) {
[735]                     System.out.println("Mods todavia existen");
[736]                     fichero1.delete();
[737]                     fichero2.delete();
[738]                  }
[739]                  if (!fichero1.exists() && fichero2.exists()) {
[740]                     System.out.println("mods borrados");
[741]                  }
[742]                  repeticiones = 24;
[743]
[744]                  state = false;
[745]                  System.out.println("Carros que llegaron al semaforo A " + semA);
[746]                  System.out.println("Carros que SE PRONOSTICARON al semaforo A " + predA);
[747]                  System.out.println("Carros que llegaron al semaforo B " + semB);
[748]                  System.out.println("Carros que SE PRONOSTICARON al semaforo B " + predB);
[749]
[750]
[751]                  //Console.println("Porcentaje de acierto "+porcentajeA+"%");
[752]                  //Console.println("Porcentaje de acierto "+porcentajeB+"%");
[753]                  semA.clear();
[754]                  semB.clear();
[755]                  predA.clear();
[756]                  predB.clear();
[757]
[758]
[759]                  cont = 0;
[760]                  listaCont = 1;
[761]                  listaContB = 1;
[762]
[763]                  if (contciclos == rep-1) {
[764]
[765]                     System.out.println("Ultima iteracion");
[766]                     System.out.println("conteo ciclos " + contciclos);
[767]                     Console.println("Last iteration");
[768]                     respaldogeneralA.close();
[769]                     respaldogeneralB.close();
[770]                     respaldogeneralAeje.close();
[771]                     respaldogeneralBeje.close();
[772]                     System.out.println("Incidencias durante ejecucion en A "+incidenciasA);
[773]                     System.out.println("Incidencias durante ejecucion en B "+incidenciasB);
[774]
[775]                     //break;
[776]                  }
[777]
[778]               }
[779]
[780]               if (state == false) {
[781]                  if (listaCont <= lista) {
[782]
[783]                     System.out.println(list);
[784]                     tRojoA = Integer.parseInt((String) list.get(listaCont -1 ));
[785]                     System.out.println("tiempo en rojo actual en A en el ciclo " + cont + " tiempo " + tRojoA);
[786]                  }
[787]                  if (listaContB <= listaB) {
[788]
[789]                     System.out.println(list2);
[790]                     tRojoB = Integer.parseInt((String) list2.get(listaContB -1 ));
[791]                     System.out.println("tiempo en rojo actual en B en el ciclo " + cont + " tiempo " + tRojoB);
[792]
[793]                  }
[794]
[795]                  carrosB = tRojoB / tCruce;
[796]
```

```
[797]                 }
[798]
[799]
[800]                 System.out.println("contadorA1                 "+
      contadorA1);
[801]                 System.out.println("contadorA2                 "+
      contadorA2);
[802]                 contadorA1 += ra1.nextInt(5);
[803]                 contadorA2 += ra2.nextInt(5);
[804]
[805]                 System.out.println("contadorA1
      "+contadorA1);
[806]                 System.out.println("contadorA2
      "+contadorA2);
[807]
[808]
[809]
[810]
[811]     //para ciclos posteriores saber cuantos numeros genero el
      random
[812]
[813]     //si el limite de simulacion no se a alcanzado
[814]     //A1 esta en rojo
[815]                 System.out.println("Carros          en          A
      "+contadorA1);
[816]                 System.out.println("Carros          en          B
      "+contadorA2);
[817]                 if (ea1 == false) {
[818]                    if (contadorA1 > levelAgreement) {  // si el
      conteo de carros es mayor que la cantidad que pueden pasar el
      SLA en un ciclo se considera que hay probabilidad de
      enbotellamientos
[819]                       try {       //escribimos en .csv la
      cantidad de vehiculos que llego
[820]                          System.out.println("A1 Rojo "+ timer +
      ", " + contadorA1);
[821]                          bw.write(timer + ", " + contadorA1 +
      ",\n");
[822]
[823]                          if (state == false) {
[824]                             contIncA++;
[825]                             respaldogeneralAeje.write(timer + ",
      " + contadorA1 + ", \n");
[826]                             System.out.println("Entre al if en A
      incidente agregado "+contIncA);
[827]                          } else {
[828]                             contEntA++;
[829]                             respaldogeneralA.write(timer + ", "
      + contadorA1 + ", \n");
[830]                             System.out.println("conteo          de
      eventos "+contEntA);
[831]                          }
[832]                          Console.println(timer + ", " +
      contadorA1);
[833]                       } catch (IOException ioe) {
[834]                       }
[835]                    }
[836]     //A1 esta en verde
[837]                 } else {
[838]                    if (state == false) {
[839]
[840]                       val = contadorA1  * tCruce;
[841]                       if (tRojoA > val) {
[842]                          carrosA = val / tCruce;
[843]                          tiempo = true;
[844]                       } else {
[845]                          carrosA = tRojoA / tCruce;
[846]                       }
[847]                       carros = carrosA;
[848]                       semA.add(contadorA1);
[849]
[850]                       System.out.println("Carro          que
      pueden pasaren A1 " + carrosA);
[851]                    }
[852]                    System.out.println("Carros "+ carros);
[853]                    System.out.println("A Verde");
[854]                    System.out.println("CARROS          EN          A
      " + contadorA1);
[855]                    contadorA1 = contadorA1 - carros;
[856]                    if (contadorA1 < 0) {
[857]                       contadorA1 = 0;
[858]                    }
[859]
[860]                    System.out.println("CARROS
      RESTANTES EN A " + contadorA1);
[861]                    slaExc = contadorA1 >= levelAgreement;
      //si hay mas carros de los que pueden pasar y lo que queda es
      mayor que el nivel de acuerdo
[862]                    listaCont++;
[863]                 }
[864]     //A2 esta en verde
[865]                 if (ea2 == true) {
[866]                    if (state == false) {
[867]                       val2 = contadorA2 * tCruce;
[868]                       if (tRojoB > val2) {
[869]                          carrosB = val2 / tCruce;
[870]                          tiempo2 = true;
[871]                       } else {
[872]                          carrosB = tRojoB / tCruce;
[873]                       }
[874]                       carros = carrosB;
[875]                       semB.add(contadorA2);
[876]                       System.out.println("Carro          que
      pueden pasar en B " + carrosB);
[877]                    }
[878]                    System.out.println("Carros "+ carros);
[879]                    System.out.println("B verde");
```

```
[880]                     System.out.println("CARROS EN B " + contadorA2);
[881]                     contadorA2 = contadorA2 - carros;
[882]                     if (contadorA2 < 0) {
[883]                        contadorA2 = 0;
[884]                     }
[885]
[886]                     System.out.println("Contador A2 "+contadorA2);
[887]                     System.out.println("CARROS RESTANTES EN B " + contadorA2);
[888]                     slaExc = contadorA2 >= levelAgreement;
[889]                     listaContB++;
[890]       //A2 esta en rojo
[891]                  } else {
[892]                     if (contadorA2 > levelAgreement) {  // si el conteo de carros es mayor que la cantidad que pueden pasar el semaforo en un ciclo se considera que hay probabilidad de enbotellamientos
[893]                        try {
[894]                           System.out.println("A2 Rojo "+ timer + ", " + contadorA2);
[895]                           bwB.write(timerB + ", " + contadorA2 + ", \n");
[896]
[897]                           if (state == false) {
[898]                              contIncB++;
[899]                              respaldogeneralBeje.write(timer + ", " + contadorA2 + ", \n");
[900]                              System.out.println("Entre al if en B incidente agregado "+contIncB);
[901]                           } else {
[902]                              contEntB++;
[903]                              respaldogeneralB.write(timer + ", " + contadorA2 + ", \n");
[904]                              System.out.println(contEntB);
[905]                           }
[906]                           Console.printline(timerB + ", " + contadorA2);
[907]
[908]                        } catch (IOException ioe) {
[909]                        }
[910]                     }
[911]
[912]                  }
[913]                  //Modificamos semaforos para siguiente ciclo
[914]                  ea1 = ea1 == false;
[915]                  ea2 = ea2 != true;
[916]                  date++;
[917]
[918]
[919]
[920]                  if (state == false) {
[921]                     if (tiempo == true) {
[922]                        timer += val;
[923]                     } else {
[924]                        timer += tRojoA;
[925]                     }
[926]                     if (tiempo2 == true) {
[927]                        timerB += val2;
[928]                     } else {
[929]                        timerB += tRojoB;
[930]                     }
[931]                     contciclos++;
[932]                  } else {
[933]                     timer += red;
[934]                     timerB += red;
[935]                  }
[936]
[937]                  cont++;
[938]                  if(cont == repeticiones && state == false){
[939]                     Console.println("End of the execution");
[940]                     Console.println("End of the execution");
[941]                  }
[942]                  System.out.println("Fin iteracion");
[943]                  dif1 = contadorA1;
[944]                  dif2 = contadorA2;
[945]
[946]
[947]               }
[948]
[949]
[950]            } catch (IOException | InterruptedException ex) {
[951]               Logger.getLogger(teisiVista.class.getName()).log(Level.SEVERE, null, ex);
[952]            }
[953]         }
[954]
[955]
[956]      }
[957]
[958]      public class predA implements Runnable {
[959]
[960]         @Override
[961]         public void run() {
[962]            try {
[963]               try {
[964]
[965]                  String path = "";
[966]                  ficheroPredA = new File(sPredA);
[967]                  bwPredA = new BufferedWriter(new FileWriter(sPredA));
[968]                  bwPredResp = new BufferedWriter(new FileWriter(sPredResp, true));
[969]
```

```
[970]                 path = "C:\\Users\\BzlZavLed\\Documents\\NetBeansProjects\\Tesis\\A.arff";
[971]     // load the data
[972]                 Instances wine = new Instances(new BufferedReader(new FileReader(path)));
[973]     // new forecaster
[974]                 WekaForecaster forecaster = new WekaForecaster();
[975]     // set the targets we want to forecast. This method calls
[976]     // setFieldsToLag() on the lag maker object for us
[977] forecaster.setFieldsToForecast("seconds,carros");
[978]                 forecaster.setBaseForecaster(new MultilayerPerceptron());
[979]                 forecaster.getTSLagMaker().setMinLag(1);
[980]                 forecaster.getTSLagMaker().setMaxLag(24); // monthly data
[981]     // add a month of the year indicator field
[982]
[983]     // build the model
[984]                 forecaster.buildForecaster(wine, System.out);
[985]                 forecaster.primeForecaster(wine);
[986]     // forecast for 12 units (months) beyond the end of the
[987]     // training data
[988] java.util.List<java.util.List<NumericPrediction>> forecast = forecaster.forecast(24, System.out);
[989]     // output the predictions. Outer list is over the steps; inner list is over
[990]     // the targets
[991]                 //int repe = Integer.parseInt(predict.getText());
[992]                 for (int i = 0; i < 24; i++) {
[993]                     java.util.List<NumericPrediction> predsAtStep = forecast.get(i);
[994]                     for (int j = 0; j < 2; j++) {
[995]                         NumericPrediction predForTarget = predsAtStep.get(j);
[996]                         double text = predForTarget.predicted();
[997]
[998]                         String text2 = String.valueOf(text);
[999]                         System.out.print("" + predForTarget.predicted() + " ");
[1000]                        //Console.printpred(text2);
[1001]                        bwPredA.write(text2);
[1002]                        bwPredA.newLine();
[1003]                        bwPredResp.write(text2);
[1004]                        bwPredResp.newLine();
[1005]                    }
[1006]                    System.out.println();
[1007]                }
[1008]                bwPredA.close();
[1009]
[1010]            } catch (Exception ex) {
[1011]            }
[1012]            FileReader ls = null;
[1013]            int lines = 0;
[1014]            int contLec = 1;
[1015]            LineNumberReader lns = null;
[1016]            ls = new FileReader(sPredA);
[1017]            lns = new LineNumberReader(ls);
[1018]            lns.skip(Long.MAX_VALUE);
[1019]            lines = lns.getLineNumber();
[1020]            System.out.println(lines);
[1021]
[1022]            BufferedReader lec;
[1023]
[1024]            lec = new BufferedReader(new FileReader(sPredA));
[1025]            String leec = lec.readLine();
[1026]
[1027]            while (contLec <= lines) {
[1028]
[1029]                try {
[1030]                    if (contLec % 2 == 0) {
[1031]                        if(leec.charAt(0)== '-'){
[1032]                        leec.replaceFirst("-", "");
[1033]                        Console.printpred(leec);
[1034]                        }else{
[1035]
[1036]                        Console.printpred(leec);
[1037]                        }
[1038]                    }
[1039]                } catch (NumberFormatException ex) {
[1040]
[1041]                }
[1042]                leec = lec.readLine();
[1043]                contLec++;
[1044]            }
[1045]        } catch (FileNotFoundException ex) {
[1046] Logger.getLogger(teisiVista.class.getName()).log(Level.SEVERE, null, ex);
[1047]        } catch (IOException ex) {
[1048] Logger.getLogger(teisiVista.class.getName()).log(Level.SEVERE, null, ex);
[1049]        }
[1050]    }
[1051] }
[1052]
[1053] public class predB implements Runnable {
[1054]
[1055]    @Override
[1056]    public void run() {
[1057]        try {
[1058]            try {
```

```
[1059]
[1060]                  if (ficheroB.exists()) {
[1061]                      String path = "";
[1062]                      ficheroPredB = new File(sPredB);
[1063]                      bwPredB = new BufferedWriter(new
      FileWriter(sPredB));
[1064]
[1065]                      path =
      "C:\\Users\\BzlZavLed\\Documents\\NetBeansProjects\\Tesis\\B
      .arff";
[1066]
[1067]      // load the data
[1068]                      Instances wine = new Instances(new
      BufferedReader(new FileReader(path)));
[1069]                      // new forecaster
[1070]                      WekaForecaster forecaster = new
      WekaForecaster();
[1071]      // set the targets we want to forecast. This method calls
[1072]      // setFieldsToLag() on the lag maker object for us
[1073]
      forecaster.setFieldsToForecast("seconds,carros");
[1074]                      forecaster.setBaseForecaster(new
      MultilayerPerceptron());
[1075]                      forecaster.getTSLagMaker().setMinLag(1);
[1076]
      forecaster.getTSLagMaker().setMaxLag(24); // monthly data
[1077]      // add a month of the year indicator field
[1078]
      forecaster.getTSLagMaker().setAddMonthOfYear(true);
[1079]                      // add a quarter of the year indicator field
[1080]
      forecaster.getTSLagMaker().setAddQuarterOfYear(true);
[1081]                      // build the model
[1082]                      forecaster.buildForecaster(wine,
      System.out);
[1083]                      forecaster.primeForecaster(wine);
[1084]
      java.util.List<java.util.List<NumericPrediction>> forecast =
      forecaster.forecast(24, System.out);
[1085]      // output the predictions. Outer list is over the steps; inner
      list is over
[1086]      // the targets
[1087]                      //int repe2 =
      Integer.parseInt(predict.getText());
[1088]                      for (int i = 0; i < 24; i++) {
[1089]                          java.util.List<NumericPrediction>
      predsAtStep = forecast.get(i);
[1090]                          for (int j = 0; j < 2; j++) {
[1091]                              NumericPrediction predForTarget =
      predsAtStep.get(j);
[1092]                              double text =
      predForTarget.predicted();
[1093]                              String text2 = String.valueOf(text);
[1094]                              System.out.print("" +
      predForTarget.predicted() + " ");
[1095]                              //Console.printpredB(text2);
[1096]                              bwPredB.write(text2);
[1097]                              bwPredB.newLine();
[1098]                              bwPredResp.write(text2);
[1099]                              bwPredResp.newLine();
[1100]                          }
[1101]                          System.out.println();
[1102]                      }
[1103]                      bwPredB.close();
[1104]
[1105]                  } else {
[1106]                      Thread.currentThread().notify();
[1107]                  }
[1108]              } catch (Exception ex) {
[1109]              }
[1110]              FileReader lsB = null;
[1111]              int linesB = 0;
[1112]              int contLecB = 1;
[1113]              LineNumberReader lnsB = null;
[1114]              lsB = new FileReader(sPredB);
[1115]              lnsB = new LineNumberReader(lsB);
[1116]              lnsB.skip(Long.MAX_VALUE);
[1117]              linesB = lnsB.getLineNumber();
[1118]              System.out.println(linesB);
[1119]
[1120]              BufferedReader lec;
[1121]
[1122]              lec = new BufferedReader(new
      FileReader(sPredB));
[1123]              String leec = lec.readLine();
[1124]
[1125]              while (contLecB <= linesB) {
[1126]
[1127]                  try {
[1128]                      if (contLecB % 2 == 0) {
[1129]                          Console.printpredB(leec);
[1130]                      }
[1131]                  } catch (NumberFormatException ex) {
[1132]
[1133]                  }
[1134]                  leec = lec.readLine();
[1135]                  contLecB++;
[1136]              }
[1137]
[1138]          } catch (FileNotFoundException ex) {
[1139]
      Logger.getLogger(teisiVista.class.getName()).log(Level.SEVER
      E, null, ex);
[1140]          } catch (IOException ex) {
[1141]
      Logger.getLogger(teisiVista.class.getName()).log(Level.SEVER
      E, null, ex);
[1142]          }
```

```
[1143]
[1144]          }
[1145]      }
[1146]
[1147]      public class monitor implements Runnable {
[1148]
[1149]          public void run() {
[1150]              int repet = tSim / red;
[1151]              Thread mod = new Thread(new modificador());
[1152]
[1153]              if (cont == repeticiones) {
[1154]
[1155]                  if (ficheroPredA.exists() && ficheroPredB.exists()) {
[1156]                      try {
[1157]                          Console.printline("Starting modifications");
[1158]                          Console.println("Starting modifications");
[1159]                          mod.start();
[1160]                          mod.join();
[1161]                      } catch (InterruptedException ex) {
[1162]  Logger.getLogger(teisiVista.class.getName()).log(Level.SEVERE, null, ex);
[1163]                      }
[1164]                  } else {
[1165]                      Console.println("Sin incidencias");
[1166]                      Console.printline("Sin incidencias");
[1167]                      Thread.currentThread().notifyAll();
[1168]                  }
[1169]              }
[1170]          }
[1171]      }
[1172]
[1173]      public class modificador implements Runnable {
[1174]
[1175]          @Override
[1176]          public void run() {
[1177]              FileReader fr = null;
[1178]              try {
[1179]                  int contL = 1;
[1180]                  int contLB = 1;
[1181]                  int line = 1;
[1182]                  int lineB = 1;
[1183]                  int reps = 0;
[1184]                  int timeRojoMod;
[1185]                  int timeRojoModB;
[1186]                  int timeCross = tCruce;
[1187]                  String mod = "modA.txt";
[1188]                  String modB = "modB.txt";
[1189]                  LineNumberReader lnr = null;
[1190]                  LineNumberReader lnr2 = null;
[1191]
[1192]                  fichero1 = new File(mod);
[1193]                  bwModA = new BufferedWriter(new FileWriter(mod));
[1194]
[1195]                  BufferedReader readerB;
[1196]                  fichero2 = new File(modB);
[1197]                  bwModB = new BufferedWriter(new FileWriter(modB));
[1198]
[1199]                  //get lines from prediction for A
[1200]                  fr = new FileReader(sPredA);
[1201]                  lnr = new LineNumberReader(fr);
[1202]                  lnr.skip(Long.MAX_VALUE);
[1203]                  line = lnr.getLineNumber();
[1204]                  System.out.println(line); //Add 1 because line index starts at 0
[1205]                  //calculate new tRojo times     for A
[1206]                  double d;
[1207]                  BufferedReader reader;
[1208]
[1209]                  try {
[1210]
[1211]                      reader = new BufferedReader(new FileReader(sPredA));
[1212]                      String leer = reader.readLine();
[1213]                      resp.write("A");
[1214]                      resp.newLine();
[1215]                      while (contL <= line) {
[1216]
[1217]                          try {
[1218]                              if (contL % 2 == 0) {
[1219]                                  d = Double.valueOf(leer);
[1220]                                  timeRojoMod = Math.abs((int) (d * tCruce));
[1221]                                  String wrt = String.valueOf(timeRojoMod);
[1222]                                  predA.add((int) d);
[1223]                                  bwModA.write(wrt);
[1224]                                  bwModA.newLine();
[1225]                                  resp.write(wrt);
[1226]                                  resp.newLine();
[1227]                                  Console.printmodA(wrt);
[1228]                              }
[1229]                          } catch (NumberFormatException ex) {
[1230]
[1231]                          }
[1232]                          leer = reader.readLine();
[1233]                          contL++;
[1234]                      }
[1235]
[1236]                  } catch (Exception ex) {
[1237]                      System.out.println(ex.getMessage());
[1238]                  }
```

```
[1239]                 FileReader frB = new FileReader(sPredB);
[1240]                 lnr2 = new LineNumberReader(frB);
[1241]                 lnr2.skip(Long.MAX_VALUE);
[1242]                 lineB = lnr.getLineNumber();
[1243]                 System.out.println(lineB);
[1244]                 //calculate new tRojo times     for B
[1245]                 double d2;
[1246]
[1247]                 try {
[1248]                    resp.write("B");
[1249]                    resp.newLine();
[1250]                    readerB = new BufferedReader(new
       FileReader(sPredB));
[1251]                    String leerB = readerB.readLine();
[1252]                    while (contLB <= lineB) {
[1253]                       try {
[1254]
[1255]                          if (contLB % 2 == 0) {
[1256]                             d2 = Double.valueOf(leerB);
[1257]                             timeRojoModB = Math.abs((int) (d2 *
       tCruce));
[1258]                             String wrtb =
       String.valueOf(timeRojoModB);
[1259]                             predB.add((int) d2);
[1260]                             bwModB.write(wrtb);
[1261]                             bwModB.newLine();
[1262]                             resp.write(wrtb);
[1263]                             resp.newLine();
[1264]                             Console.printmodB(wrtb);
[1265]                          }
[1266]                       } catch (NumberFormatException ex) {
[1267]                          System.out.println("Not   Double   '   "   +
       leerB + " '");
[1268]                       }
[1269]
[1270]                       leerB = readerB.readLine();
[1271]                       contLB++;
[1272]                    }
[1273]
[1274]                 } catch (Exception ex) {
[1275]                    System.out.println(ex.getMessage());
[1276]                 }
[1277]                 bwModA.close();
[1278]                 bwModB.close();
[1279]
[1280]                 lnr.close();
[1281]                 lnr2.close();
[1282]                 ficheroPredA.delete();
[1283]                 ficheroPredB.delete();
[1284]
[1285]              } catch (FileNotFoundException ex) {
[1286]
       Logger.getLogger(teisiVista.class.getName()).log(Level.SEVER
       E, null, ex);
[1287]              } catch (IOException ex) {
[1288]
       Logger.getLogger(teisiVista.class.getName()).log(Level.SEVER
       E, null, ex);
[1289]              } finally {
[1290]                 try {
[1291]                    fr.close();
[1292]                 } catch (IOException ex) {
[1293]
       Logger.getLogger(teisiVista.class.getName()).log(Level.SEVER
       E, null, ex);
[1294]                 }
[1295]              }
[1296]           }
[1297]        }
[1298]
[1299]
[1300]
[1301]
       }
```