2014 International Conference on Advances in ICT for Emerging Regions (ICTer): 091 - 095

Genetic Programming Tuned Fuzzy Controlled Traffic Light System

T.D.N.D. Padmasiri[#1], D.N. Ranasinghe[*2]

[#]*IFS R and D International (pvt) Ltd. Sri Lanka*

[1]nilusha.padmasiri@ifsworld.com

[*]*Department of Computation and Intelligent Systems,*

University of Colombo School of Computing

[2]dnr@ucsc.cmb.ac.lk

***Abstract* — A blend of fuzzy logic and genetic programming is used in this research to achieve a single fine-tuned fuzzy rule, upon giving hundreds of fuzzy rules as the input.. The system has Poisson arrival rate of vehicles, and decisions are taken to alter the sequence of lights based on the queue lengths of the lanes. The traffic simulator handles routing of vehicles in a single four-leg intersection with left and right turns. The fuzzy logic traffic controller system is used to generate the simulation data to feed the genetic programming system. The genetic programming system then creates an optimum fuzzy rule. This fine-tuned fuzzy rule is proven to be qualitatively better with respect to the mean square queue length and its error of the total system at any given point of time.**

***Keywords*— fuzzy logic, genetic programming, traffic controller system.**

I. INTRODUCTION

Traffic Congestion has become a major discussion point today, in any developed or developing cities. Building new roadways or widening existing roadways may be the most straight forward answers on this matter. But managing the traffic in roads, have major impact on the traffic congestion. The present traffic controller systems are mostly pre-timed and cannot handle dynamic nature of the traffic. However, if the traffic controller system can handle the dynamic queue lengths and vehicle arrival rates at a junction, the traffic flow is expected to be smoother than in the present pre-timed traffic controlling situation.

Fuzzy logic seems to be a better candidate in this regard, as it supports pervasive uncertainty and vagueness in real world decision making situations. However in such implementations, hundreds of fuzzy rules are introduced to cater to different situations and combinations. This in a way is, applying subject expertise into logic and trying to retrieve the best answer in a question. The present situation simply depends on the activated fuzzy rules upon given set of inputs and taking the output using fuzzy inference engine.

Genetic programming on the other hand is being used as a technique to automatically discover computer programs using principles of Darwinian evolution. If, we could go beyond the expert knowledge and simple fuzzy rules, to arrive at a genetically improved rule, we should be getting better results. Hence, the blend of fuzzy logic and genetic programming should provide promising results of better accuracy, effectiveness and efficiency in the system they are embedded.

This research attempts to reduce the number of fuzzy rules included in a system by fine tuning them using genetic programming. The resulting fuzzy rule set would include only very few rules or a single rule, but which are/is capable of handling all most all the scenarios, which implies that the accuracy of the system should also be increased with this.

Section 1 describes the background, motivation, objectives of the research and the scope of the research in a nutshell. State of art of usage of fuzzy logic in traffic controller systems, instances where fine tuning of fuzzy rules done using genetic programming are in detailed described in section 2. Section 3 is reserved for the design and implementation details of the system and Section 4 to evaluate the system. Finally section 5 concludes with future work.

II. RELATED WORK

The research done in this paper is an extension to existing research on fuzzy logic traffic controller systems. There are vast amount of research done such as [4], [3], [8], [1], [5], [10], [11], [9], [7] on intelligent traffic controller system which has the capability of mimicking human intelligence for controlling traffic lights. The 48 rules introduced by [1] are taken as the basis for this paper.

The research in [2] is aimed at showing that Genetic Programming can be used to generate and evaluate fuzzy logic statements. The application area is financial trading. The rule set generation has being performed using Koza's 'ramped half-and-half' method. However after 10 generations, the researchers were only able to get a rule which yielded a profit of 62.2%.

In addition, Hugh Mallinson and Peter Bentley in [6] had done a research on the use of a Hybrid Fuzzy-Genetic Programming system to discover patterns in large databases. This is a similar case for this research. There the application was on a breast cancer database and the experiment results has shown that the system classify data correctly 95%.

III. METHODOLOGY

In our case, the fuzzy system modelled is, of a traffic controller system for a four-leg intersection. The simulation takes input data queue length, arrival rate and out put data to say whether to extend the current green signal or not, using this model. These simulated data is then fed to the genetic programming tool as the third step. The genetic programming system will be deriving a final rule which can be used as the fine tuned rule and this rule is compared with the initial rule

978-1-4799-7732-1/14/$31.00

set which was used to derive the final rule, under same traffic conditions. Without loss of generality, the topology for the traffic controller system is a single four-leg intersection with left and right turns. It is assumed a Poisson arrival rate of vehicles per lane per second. The result of the improved traffic controlling logic is expected provide a reduction in average queue length of the total system.

The system is created based on the 48 fuzzy rules used by [1]. The design is as such, it is expected to take hundreds of fuzzy rules and simulated data as input and come up with few number of rules or a single rule which has higher accuracy in deciding whether to extend the green signal or terminate.

The fuzzy system used in this project is developed based on the research done in [1]. The traffic flow; the project on [1] modelled, is shown in Figure 1. This considers a left hand driving set up.

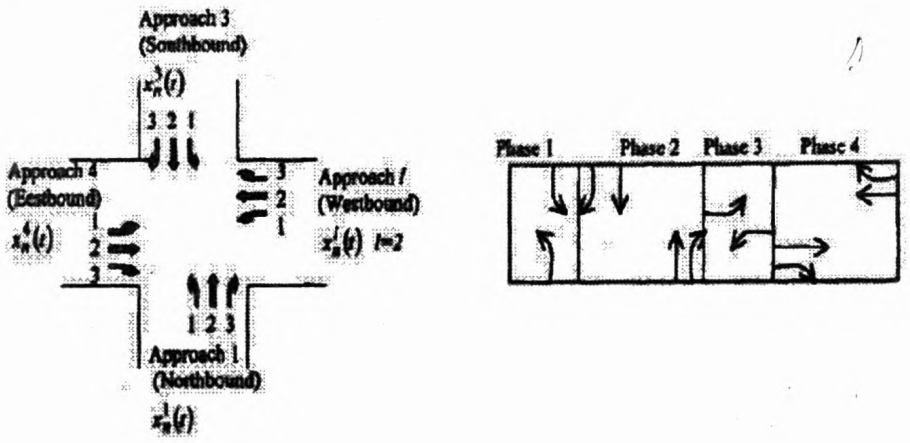

Fig. 1 The Traffic Flow modelled by the rules in research [1]

Each phase in this model is a period of time where the traffic flows towards the directions marked in the figure. The four phases will be covering all the traffic flow towards the junction from four directions. We also assume there is no dynamic phase change. Phase extension or change will be determined by the fuzzy logic rules.

The starting point is a simulation of single traffic junction with 4 roads connected. The traffic controller system is modelled using following parameters as inputs and out puts.

Inputs:

AR - Average arrival rate on lanes with the current green, in vehicles/second/lane

QC - Average queue length on the lanes served by the current green, in vehicles/lane

QN - Average queue length on lanes with red which may receive green in the next phase, in vehicles/lane

Output:

Control - Extend Or Terminate the current green duration

The tool XFuzzy is used to model the system. The membership functions are defined according to the parameters. The inputs are modelled as trapezoid and output as bell shaped. All the membership functions used are shown in Figure 2. The inference is done using the default set up provided by XFuzzy as in [12].

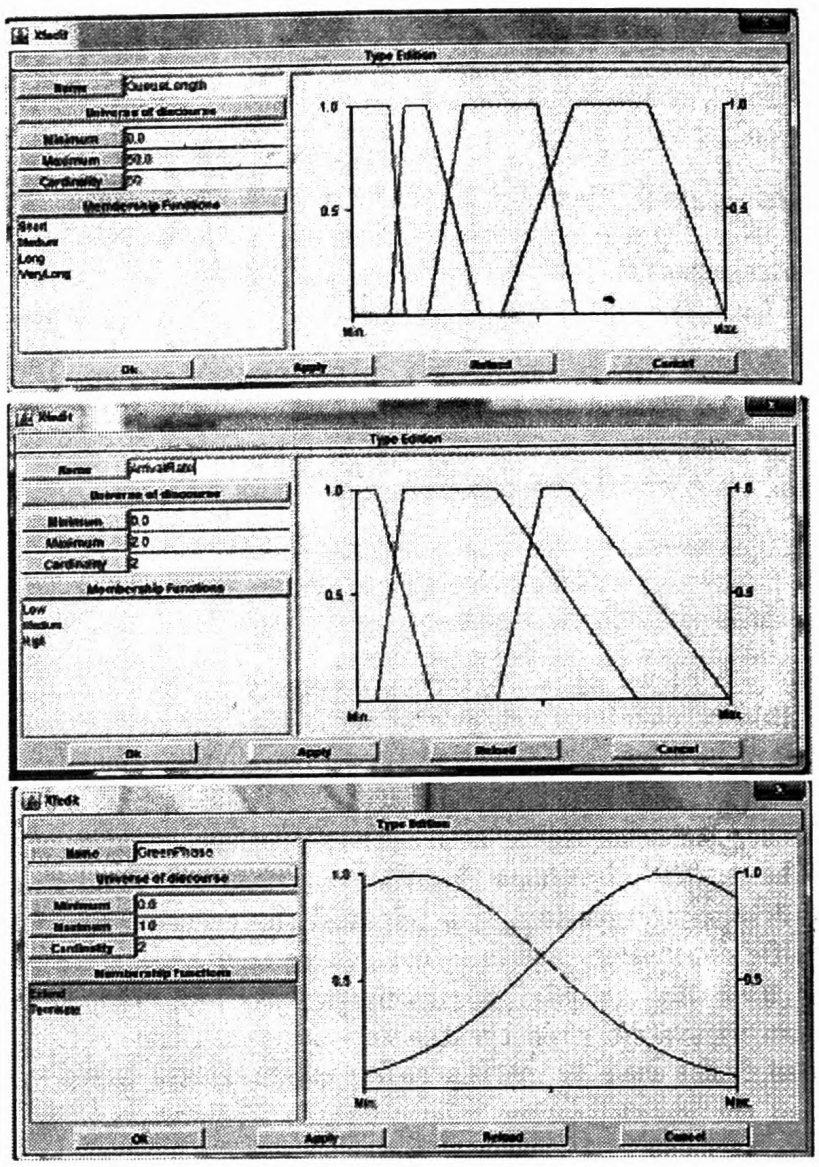

Fig. 2 The Membership functions used

The implementation of the system is done using java. The XFuzzy tool is able to generate java code for the fuzzy inference engine.

A simulation logic had to be written, to be used in simulation, as required by XFuzzy tool. The plant model required by the tool is a class file which would be generated with the simulation logic written. The simulation tool XFSim provides data in log files as per developers wish.

The generated data from simulation will then be fed to the Genetic Programming Tool. There are many genetic programming tools such as GPLAB, GPTIPS, GPalta, GPdotNet, EpochX, and AForge.NET. However GPalta was chosen to be used in the project, as it was open source, coded in java and facilitate the user with a mini user guide.

GPalta requires input and output sets of data to be feeded to the system. This is done simply using 02 text files. The black box then be creating an equation to cater the data sets provided. The genetic programming tool has provided the flexibility to run any amount of generations, the user prefers.

Figure 3 captures details provided by GPalta after 1001 generations on a given data set.

The best fitness here refers to the fitness of the generated rule when compared to the input and output data sets. The number of nodes refers to the nodes getting created in the evolution process. We can determine a number of generations to be created. The evolution will be happening until the number of specified generations are created, when you press 'Go N'.

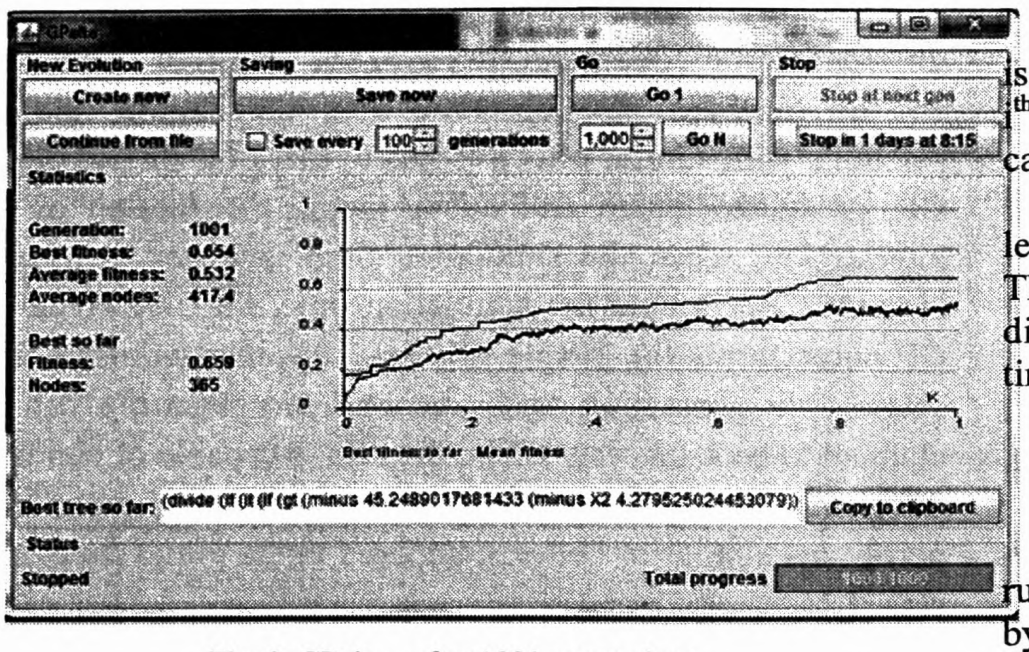

Fig. 3 GPalta - after 1001 generations

The Best Tree so far given by the genetic programming is a single rule which we expected as the final outcome of the process. This is a rule written in Prefix notation. Appendix A includes 02 such rules created.

It is observed that, there is no such requirement to provide huge amount of data for GPalta to come up with a 'good' rule. Also it is evident that the fitness value provided by GPalta cannot be taken as a validation methodology for the accuracy of the final rule. At times, even though the fitness calculated by GPalta is as high as 79% after thousand generations, the rule happened to provide un-acceptable results in the test phase and rules with 50% of average fitness yielded better results.

The system quality is measured based on running average for each queue size. The mean square error for queue length of the total system at any point of time is calculated with the same set up for the following 2 instances.

1. with traffic control done using the simple rule set from [1]

2. with traffic control done using the fine tuned single rule from genetic programming

The traffic signals are assumed to behave as follows:

- Minimum Green Phase provided at each phase change: 10 seconds
- Green Phase extension time for each run of the rule for extend or terminate the current green: 5 seconds
- Maximum Green Phase allowed at one stretch, extending the Green Phase several times: 60 seconds
- The next phase will always be same as shown in figure 1 and no dynamic phase changes are implemented.

The running average is calculated using the following equation and α set to 0.5.

$\bar{X} = \alpha_i q_i + (1-\alpha)\bar{X}$ where $0<\alpha<1$ ------ (1)

Here q_i denotes queue length of the i^{th} queue. Running average is calculated for time t_0 and mean square error is calculated after δt times after that.

Mean square error of all queue lengths is calculated using the following equation :

$\sqrt{(\sum(q_i - \bar{X})^2) /N}$ ----- (2)

Here N denotes the total number of queues and summation is done for i=0 to i=N, where qi denotes queue length of the i^{th} queue at the δt times after the running average is calculated.

The objective was to make a comparison of average queue lengths of the two systems under the same traffic conditions. The use of mean square error allows us to compare the distribution of queue lengths in the intersection at a point of time under the same traffic conditions.

IV. EVALUATION

The 48 rules fed to the fuzzy system represent hundreds of rules that is possible in a design. The compact rule generated by the GPalta is compared with the rules of XFuzzy using a simulator. Interestingly some simple rules provided by GPalta provided similar results to the initial fuzzy rule set. To evaluate the system quality, we calculated the running average for each queue and calculated the Mean Square Error for queue length of the total system.

Figure 4 shows mean square error of queue length when the same traffic conditions are faced by single rule traffic control and simple rule traffic control.

Here many different, practically possible set ups are provided and for each set up, the single rule control and simple multiple rule controls are enforced in separate instances. The X axis represents the simulation time period and Y axis shows the mean square error resulted by the two systems under same conditions. It is evident that the system with rule fine tuned via Genetic Programming System provides a majority of lower mean square error on queue length instances. When the simulation time is of a lower value, there is no much difference between the systems. But as simulation time increases the fine tuned rule outperformed the simple fuzzy rule set.

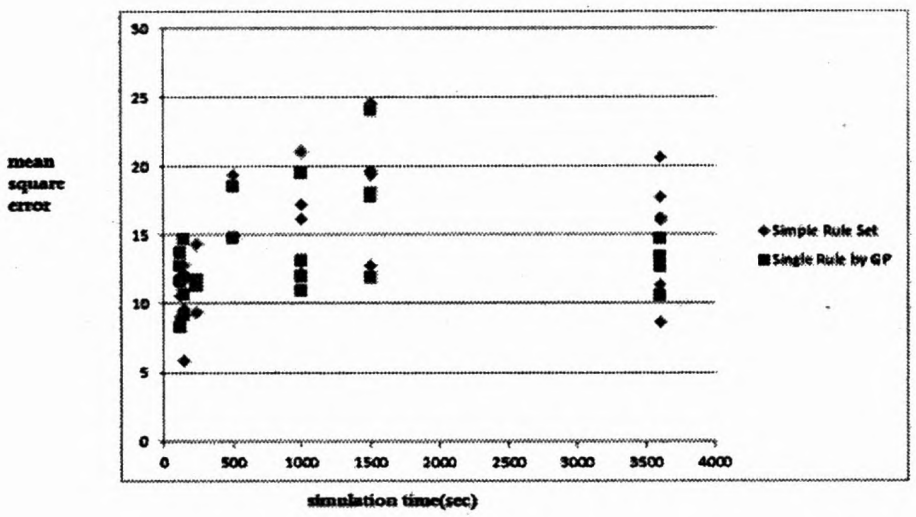

Fig. 4 Mean Square Errors with time in different traffic conditions

Further testing was done to compare the behaviour of the rules, when there is a considerably longer queue in one of the lanes. The same setup of queue length and arrival rate, run for different simulation time and the result is summarized in figure 5.

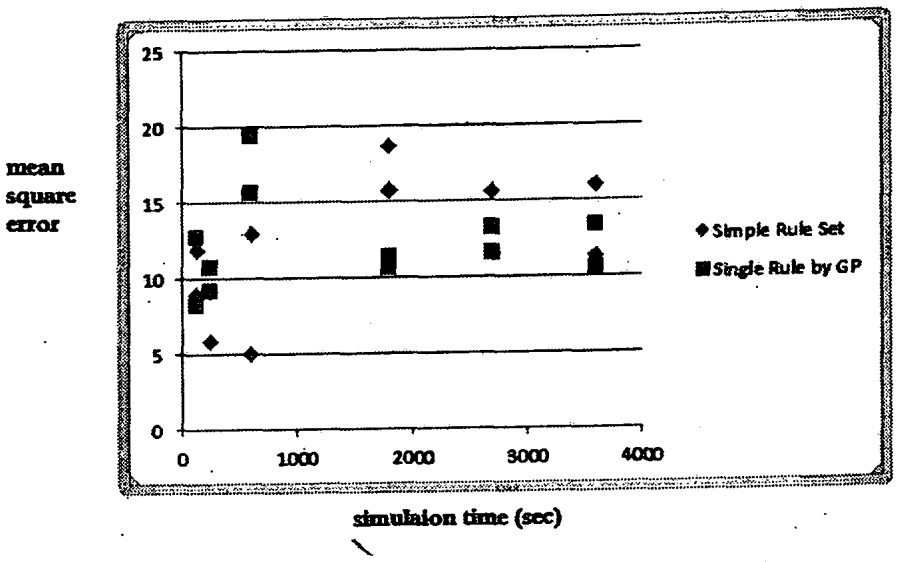

Fig. 5 Mean Square Errors with time when there is a long queue at the start

Another interesting set up is with heavy traffic volume created using longer queue lengths in most of the queues. The result is shown in figure 6. A huge traffic was generated using high arrival rates and the output is included in figure 7.

Analysing all 4 test scenarios we can conclude that, the simple rule set performs with similar effectiveness to that of a single rule by GP, on queue length and queue length distribution of the intersection, when the simulation limit is lower than 1000 seconds.

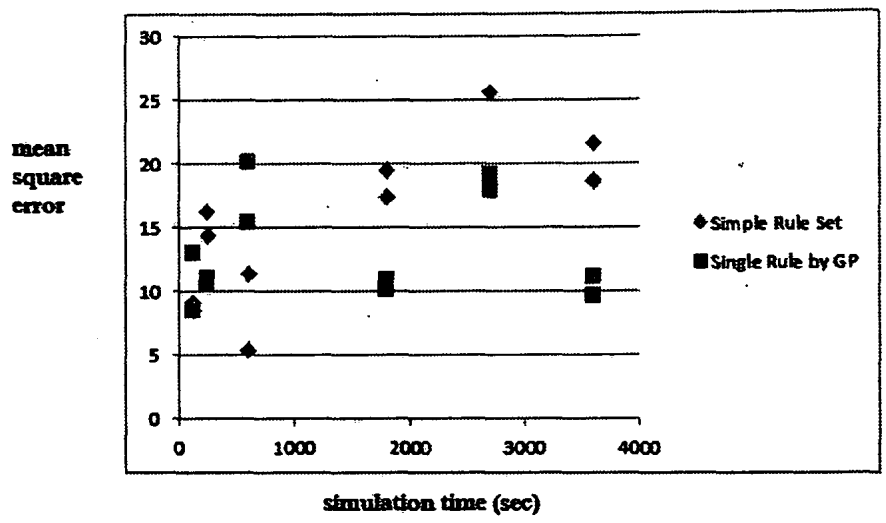

Fig. 6 Mean Square Errors with time under heavy traffic volume due to longer queue lengths

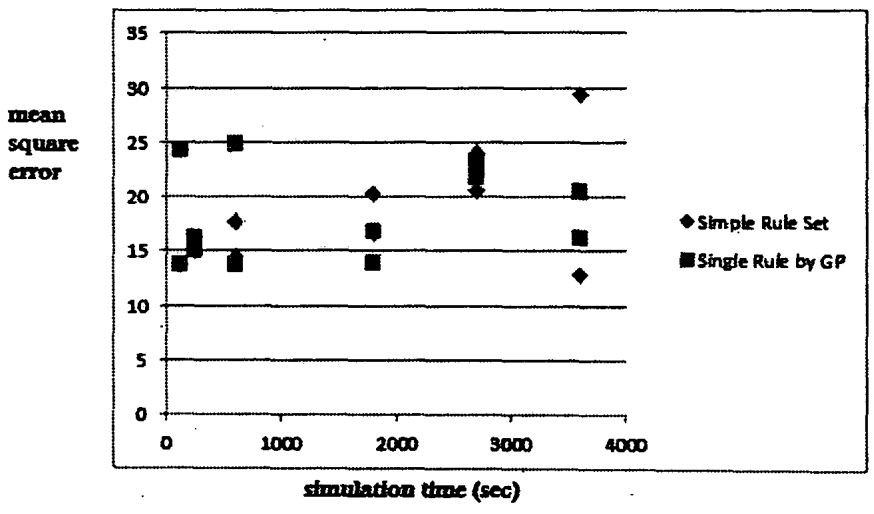

Fig. 7 Mean Square Errors with time under heavy traffic volume due to heavy arrival rates

But longer the time the system has to operate, the single rule by GP outperforms the simple rule set.

There is research such as [11] where a dynamic phase change is applied. However we predict that still, the fine-tuned rule will perform better, as it is observed that the single rule by GP seemed to be considering the overall situation.

V. Conclusions

The stability of the proposed solution is analysed under different traffic conditions based on the length of the queues and arrival rates. We observed that, the simple rule set performs with similar effectiveness to that of a single rule by GP, on queue length and queue length distribution of the intersection, when the simulation time is lower than 1000 seconds. But longer the simulation time, the single rule by GP outperforms the simple rule set. In other words, a final single rule generation using genetic programming can be claimed to be a success, where the rule is capable of handling different traffic conditions. In general, the fine-tuned rule performed similar or better, compared to the initial fuzzy rule set.

Appendix A - Rules Created

(divide (if (eq X3 2.403633507237913) (if true X3 -89.25773770100403) (plus X3
46.565688854881 81)) 51.49981998136684)

One of better performed rules (The one used in all validation tests):

(times X3 (if (eq (divide (plus (plus X1 (plus X2 (times X3 X2))) 36.51297910106524) (times
X1 (plus (if (eq (plus X3 X2) X1) (plus 22.595394769652913 X3) (times X3
22.595394769652913)) X1))) (times (divide (if (lt (times X3 X1) (times X3 X2)) (plus X2
14.476306911052035) (plus X2 (times X3 X2))) X1) (plus (plus (plus 22.595394769652913
X1) (times (times X1 X1) (divide X2 X2))) (divide (if (lt (divide X2 X2) (minus X2 X1)) X1
(plus X3 X2)) (times X3 X3))))) (plus (if (eq (times X3 (plus (times X3 X2) X2)) (plus (plus
X1 (times X3 X1)) (plus (times X3 X1) (plus X2 14.476306911052035)))) (plus (divide (if (lt
(times X3 X1) (divide X2 X2)) (plus (times X3 X2) 14.476306911052035) (plus X2 (times X3
X1))) X1) (plus 22.595394769652913 (times (plus 22.595394769652913
14.476306911052035) (times X3 X1)))) (plus (divide (if (lt (plus 14.476306911052035
14.476306911052035) X3) (plus 22.595394769652913 X3) -36.05099267517171) (plus
(times X3 (plus X3 X2)) (times X3 22.595394769652913))) (if (lt (plus X3 X2) (plus X3 X2))
(times (plus 22.595394769652913 X1) X1) (minus (plus 22.595394769652913 X1) X1)))) (if
(eq (times (times X3 X1) (plus X2 (plus X2 (times X3 X1)))) (divide (times (divide X2 X2)
(times X3 X2)) (times (minus X2 X1) (plus (plus X3 X2) (times X3 22.595394769652913)))))
(plus X1 (if (lt (divide X2 X2) (times X1 X1)) X1 (times (times X3 (times X3 X2)) (times X3
(times X3 22.595394769652913))))) (times X1 (times (times X3 X1) (plus X2 (times X3
X1)))))) (plus (divide (plus (if (lt (plus 14.476306911052035 14.476306911052035) (plus
22.595394769652913 X1)) X2 (minus X2 X1)) X2) (times (plus X3 X3) (plus (if (lt (plus X2

14.476306911052035) (times X1 X1)) (plus (plus X3 X2) (times X3 22.595394769652913))
(minus X2 (plus X3 X1))) X1))) (if (eq (minus X2 X1) (if (eq (plus X2 (divide X3 X3)) X1)
(times (plus 22.595394769652913 X1) (times (times X3 X2) 14.476306911052035)) (divide
X2 X2))) (plus (divide (if (eq (plus X3 X2) X1) 22.595394769652913 -36.05099267517171)
26
X1) (plus (plus (times (plus 22.595394769652913 X1) (times X3 X2)) 14.476306911052035)
(times (plus (plus X3 X2) (times X3 22.595394769652913)) (times (plus
22.595394769652913 14.476306911052035) (times X3 X2))))) (plus (divide (if (lt (plus
14.476306911052035 14.476306911052035) (plus 22.595394769652913 X1)) (plus
22.595394769652913 X1) -36.05099267517171) (times (times X3 X1) (plus (plus X3 X2)
(times X3 22.595394769652913)))) (times X3 (plus X2 (minus X2 6.506936936579592))))))))

REFERENCES

[1] A SEMINAR REPORT ON TRAFFIC SIGNAL CONTROL USING FUZZY LOGIC. [Online] Available: http://www.scribd.com/doc/46862640/Traffic-Signal-Control-Using-Fuzzy-Logic.

[2] D. Burkhardt & O. Adjei Andrew N. Edmonds. ``Genetic Programming of Fuzzy Logic Production Rules with Application to Financial Trading". In: (1995).

[3] M. Mahmood I. N. Askerzade. ``Control the Extension Time of Traffic Light in Single Junction by using Fuzzy Logic". In: International Journal of Electrical & Computer Sciences IJECS-IJENS 10.02 (April 2010).

[4] Rubiyah Yusof Kok Khiang Tan Marzuki Khalid. ``Intelligent Traffic Light Control By Fuzzy Logic". In: Malaysian Journal of Computer Science 9.2 (Dec. 1996), pp. 29–35.

[5] Girija H. Kulkarni and Poorva G. Waingankar. ``Fuzzy Logic Based Traffic Light Controller". In: Second International Conference on Industrial and Information Systems, ICIIS 2007, Sri Lanka (Aug. 2007), pp. 08–11.

[6] Hugh Mallinson and Peter Bentley. *"Evolving Fuzzy Rules for Pattern Classification"*. In: *Computational Integration for Modelling, Control and Automation '99* 55 (Feb. 1999), pp. 184–191

[7] Rubiyah Yusof Marzuki Khalid See Chin Liang. ``Control of a Complex Traffic Junction using Fuzzy Inference". In: 5th Asian Control Conference 3 (Jul. 2004), pp. 1544 - 1551

[8] *Nasir ALI Muhammad ABBAS M. Saleem KHAN and Syed FAZIL. ``Fuzzy LogicBased Autonomous Traffic Control System [Sensors & Transducers (Canada)]". In: 2012International Frequency Sensor Association (Mar. 2012).*

[9] Onibere Emmanuel Amano Osigwe Uchenna Chinyere Oladipo Onaolapo Francisca. ``DE- SIGN AND SIMULATION OF AN INTELLIGENT TRAFFIC CONTROL SYSTEM". In: International Journal of Advances in Engineering & Technology 1.5 (Nov. 2011), pp. 47–57.

[10] Mohammad A. Tahaa and Laheeb Ibrahimb. ``Traffic Simulation System Based on Fuzzy Logic". In: Conference Organized by Missouri University of Science and Technology - Washington D.C. (2012), pp. 356–360.

[11] Mohammad Hossein Fazel Zarandi and Shabnam Rezapour. ``A Fuzzy Signal Controller for Isolated Intersections". In: Journal of Uncertain Systems 3.3 (2009), pp. 174–182.

[12] xfuzzy-team@imse-cnm.csic.es. FUZZY LOGIC DESIGN TOOLS IMSE-CNM 2012