Journal of Computer Science

Original Research Paper

A Fuzzy Logic-Based Smart Traffic Management Systems

[1]Adel Abdallah Abdou, [1]Mohamed Hassan Farrag and [2,3]A. S. Tolba

[1]Department of Information Systems, Faculty of Computers and Information, Fayoum University, Fayoum, Egypt
[2]Department of Computer Science, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
[3]New Heliopolis Engineering Institute, Cairo, Egypt

Article history
Received: 12-08-2022
Revised: 12-09-2022
Accepted: 22-09-2022

Corresponding Author:
Adel Abdallah Abdou
Department of Information Systems, Faculty of Computers and Information, Fayoum University, Fayoum, Egypt
Email: aa4367@fayoum.edu.eg

Abstract: Traffic congestion is a serious problem in many developing countries like Egypt. In the presence of the traditional traffic light system, it creates health risks, wastes time and fuel, and pollutes the air. Therefore, there is a big need for a smart traffic management system. This study contributes in solving this problem by introducing an artificial intelligence-based smart traffic light system using fuzzy logic to ensure the smooth flow of traffic in cities. Research results indicate that smart traffic light management is very crucial in smart cities to reduce their carbon footprint to save the world and save energy. Other possible application could be implemented with minor system upgrades such as Detection and Management of traffic Congestion, Automatic Billing of Toll Charges, Automatic detection of speed limit Violation, Route planning, Intelligent Internet of Vehicles, and Prevention of Road Accidents. Four traffic light approaches (fixed-time, smart-time, fixed-time-fuzzy logic, and smart-time-fuzzy-logic) are developed with different eight traffic scenarios. The four approaches are applied with two tools, the first tool is a real Marquette which consists of 4 roads with only one or two vehicles. The second is a simulation software which consists of 4 roads with more than 100 vehicles through 10 min. Many sensors such as (IR sensors, rain sensors, and LEDs to control traffic light) are used to collect data and then the fuzzy algorithm is used to ensure the smooth flow of traffic in cities. The traffic data acquired from the vehicles is fed into the proposed model to maximize the duration of the green light based on the road state. According to experimental results, the proposed smart fuzzy technique simulation software reduced the average waiting time of the vehicles from 769 to 289 sec in heavy rain condition. It also reduced the total time for all vehicles from 2490 to 1154 sec.

Keywords: Traffic Congestion, Fuzzy Logic, Smart Traffic System, Optimization, Arduino Mega

Introduction

Many major cities around the world can face significant traffic congestion. The traditional traffic light system is based on a fixed time concept that is assigned to each side of the intersection and does not fluctuate in response to traffic congestion at specific intersections. This results in wasted time and fuel usage and undesired traffic jams. Traffic congestion leads to high levels of pollution affecting the health of local residents, passengers, and animals. Most of this is due to the poor timing cycle of the countdown timers. The timer was set equally for both high-intensity lanes and low-intensity lanes. This causes unnecessary parking in the high-density lane even if other lanes are empty. The straight forward solution to the matter is to regulate traffic signals consistent with traffic density based on fuzzy logic control Nigam *et al*. (2022).

The main purpose of this study is to create a smart system to control the road traffic based on the density and rain condition information using fuzzy logic Tomar *et al.* (2022) technique to obtain the perfect time for every junction. The proposed model uses two types of sensors to collect road data. The IR Agarwal and Rai (2022) sensors arranged at four lanes to detect presence of the vehicles and rain sensor Yang *et al*. (2022) to presence the rain condition. The sensors data is given to ARDUINO MEGA Hlavacek and Gotthans (2022)

which processes data to control traffic signals accordingly to road status based on fuzzy logic control. Seven segments display-based count timer are used. Initially, when sensors aren't obstructed by any vehicles, count time is zero and the light status is red. when sensors are obstructed by vehicles, count time is start, and the light change to the green. Thus, this provides effective solution for traffic jam in busy roads. There are four main traffic light approaches with eight traffic scenarios presented in this study.

Fixed-Time approach: In this system, the green light is given with a fixed time for each road that does not change due to the intensity of the road or the rain conditions.

Smart-Time approach: In this system, the traffic light time changes based on the numerical density of each road without taking into account the weather conditions such as rain and it opens the densest road first, then the least, etc. It is converted to a red light in the absence of vehicles.

Fixed-Time-Fuzzy-Logic approach: In this system, the traffic light time changes based on two factors. First, rain condition is (No, Light, Medium, High). The second is the numerical density of each road. The routes change sequentially that mean it start from route A until route D and repeat the turn again. If any route has no cars, the controller will be skipped to the next route. Also, if there are no cars in all routes the light will become red. Counter time is flexible change depend on the number of care and the rain value.

Smart-Time-Fuzzy-Logic approach: In this system, the traffic light time changes based on two factors. First, the rain condition is (No, Light, Medium, or High). The second is the numerical density of each road. It also opens the densest road first, then the least, and so on like smart time approaches but it gives more time for each road based on the rain condition. It is converted to a red light in the absence of vehicles.

The proposed system considered not only the density of vehicles on the roads but also the weather conditions. The proposed smart system contributes to optimally solving the problem by:

- Increase traffic rates accuracy
- Decrease the average waiting time that leads to fuel consumption
- Provide more stability
- Increase the degree of reliability whenever applied in the traffic system

This section reviews the recent research studies of traffic light systems. For example, Kammoun *et al*. (2014) suggested a hierarchical fuzzy model and ant colony behavior-based adaptive multiagent system. By including an adaptive vehicle route guidance system, this system enables effective traffic adjustment in response to real-time changes in road networks.

Ahmad *et al*. (2014). Define the Earliest Deadline First (EDF) algorithm to alleviate traffic congestion in metropolitan areas and manage to reduce the average waiting time. Adewale *et al*. (2018) provided a fuzzy-based model and achieved an average vehicle speed performance of 8%. Adewale *et al*. (2018) proposed a smart traffic system based on fuzzy logic control systems by using IR sensor and siren detection system using for emergency and security road users. This system given priority for high-density road with suitable time to increase and pedestrians available. This system decreases junction and delay of traffic. Studyana *et al*. (2018) proposed another model based on fuzzy logic for traffic signal intersection in Indonesia highway capacity. Fuzzy logic system analysis based on linguistic variables that obtained from signal intersection from the 2015 traffic research survey in Bandung. This system decreased the number of vehicle queues compare with intersection with a fixed-time signal. Elda *et al*. (2019) proposed a new traffic light model using Fuzzy Logic. This system determined the time of the green signal based on a variable density. This study applied at Tirana which is the capital of Albania. This model succeeded to minimize the congestion at an intersection. Mualifah and Abadi (2019) proposed another fuzzy logic control system to enhancement on easy traffic. This system applied at Sultan Agung Street Yogyakarta. This system is based on three in puts variables like number of cars, number of motorcycles, and the length of the queue. Traffic light is varied for each traffic conditions. They achieved high results compared with traditional fixed time. Patel and Rohilla (2020) proposed an adaptive control traffic system using Arduino mega and IR sensor. The model has been used to measure only the density of vehicles the road lanes and give green signal based on lane density to reduce traffic congestion. By transmit the data from detected vehicle by IR sensors to controller and mega give priority to lane density. They didn't consider the rain condition. From the previous studied, it can be concluded that many techniques were used to address the traffic signaling challenges. The green time management, cycle time improvements, and optimization of the average speed of the vehicle are the most common techniques. Also, these studies didn't consider the conditions and weather factors such as rain and fog.

Methodology and System Design

Two types of traffic light tools are being studied some of using the fuzzy logic algorithm. The first is a Maquette of real traffic signal intersection with four roads, each containing one or two vehicles as shown in Fig. 1. The

second is a simulation software to simulate a complex traffic light intersection with four roads and more than 100 vehicles through 10 min. The Architecture design of traffic road is shown in Fig. 2.

In the proposed model environment, there are four routes with only one intersection. The routes will be given the following symbols: A, B, C, and D. The sensors used for each route are two Infrared sensors and one led indicator (Green, Red). Rain sensor to detect the level of rain. It is also found in the intersection one cathode seven segment display and ARDUINO MEGA and connected wires are created in the maquette. The software used in the experiments are Arduino IDE (Integrated Development Environment). The experimental setup of the proposed models is to prepare and assemble the hardware components and write a program file to control the smart traffic signal through the intersection and system that we are designing as a prototype system for the real system**.** Adoni *et al*. (2022) proposed, a scalable and real-time intelligent transportation system based on a big data framework. Their system allows for the use of existing data from road sensors to better understand traffic flow and traveler behavior and increase road network performance. The transportation system is designed to process large-scale stream data to analyze traffic events such as incidents, crashes, and congestion.

Fuzzy Logic System (FLS)

Fuzziness refers to the degree of uncertainty in concept or a system. Control models are easily implemented using fuzzy logic, providing effective solutions based on ambiguous and confusing information Xiao (2022). For each road at an intersection, the created fuzzy logic model takes two inputs and generates an optimized green time duration. Figure 3 shows the operations of fuzzy logic. Based on proposed model, the proposed fuzzy model uses different rules to determine the duration of the green light.

System Design and Simulation Software

A. System Maquette Prototype

Four roads with one junction intersection models were built to be consistent with the Egypt traffic system. These roads are similar to the real roads in Egypt. This prototype collect data from eight IR sensors arranged on 4 roads. Each road contains two sensors and all roads contain only one rain sensor. Rather than requiring a physical press of the reset button before an upload, the Mega 2560 is designed in a way that allows it to be reset by software running on a connected computer.

The Arduino Software (IDE) uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. The Mega 2560 board has a number of facilities for communicating with a computer. The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The traffic light system consists of five important components. The first part is the controller that represents the Brain of the Traffic System that controls the selection and timing of the traffic movements according to the different requirements of the traffic signal consisting of red and green lights. Red for stopping vehicle and green for pass the vehicle.

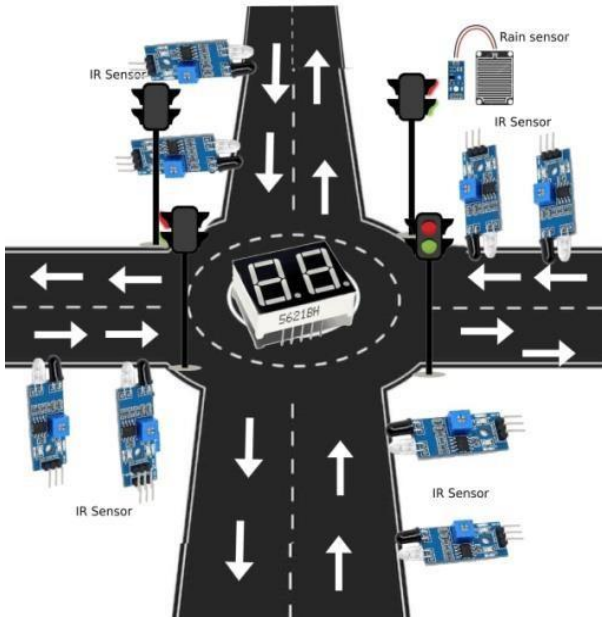

Fig. 1: Maquette of real traffic road architecture

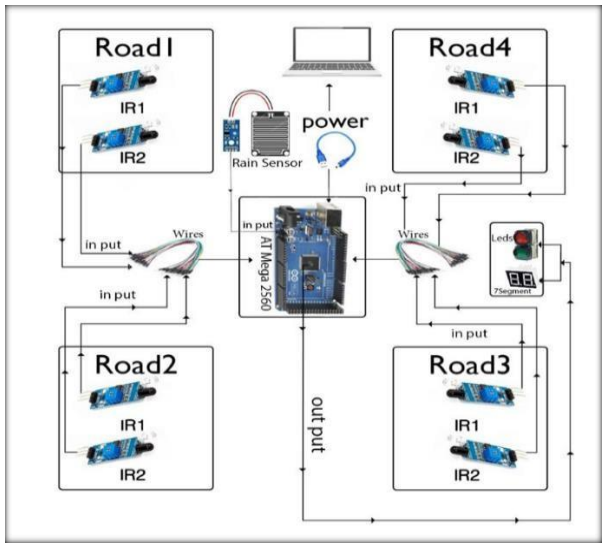

Fig. 2: Architecture of traffic road system

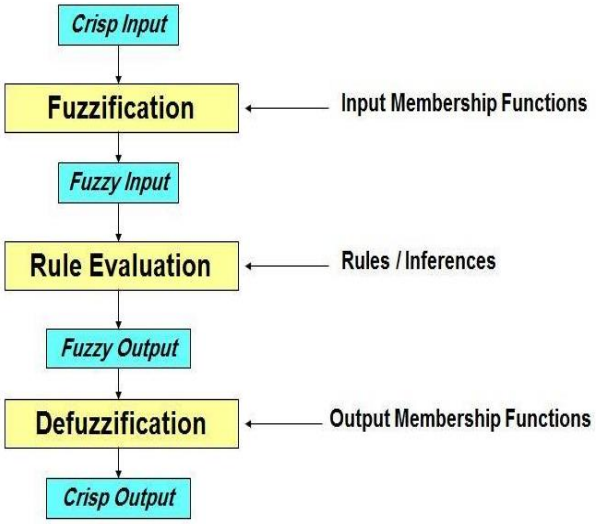

Fig. 3: Operations of the fuzzy logic system

The third part is the IR sensor. A sensor or detector is a device to indicate the presence of vehicles the fourth is the rain sensor that defines the condition rain if any. the five parts is the seven-segment display recorded by the control unit Sensors. The second part is the LEDs. These usually for count to turn on the signal and close another.

B. System Simulation

This system collect data from only the simulation software and rain sensor. This software simulates the maquette by opening and closing traffic lights, choosing roads and meter timings, and sensing rain through the rain sensor. Also, it can print the number of vehicles in all roads, the value of rain sensor in the case of raining and also printing the road that will be open, and the timings of the meter on the serial monitor of IDE.

Proposed Models

Each Scenario was analyzed individually in this study and the ideal green light time was predicted using a generated fuzzy logic model. For ten minutes, 100 vehicle, and value of rain sensor, the simulated models were tested. In maquette use case, the proposed fuzzy model was used with data from IR and rain sensors. As a result, we aim to reduce the total time, average vehicle waiting times, and average queue length at intersections.

Maquette-Based Traffic Light System

This method includes four models for the four approaches (Fixed-Time, Smart-Time, Fixed-Time Fuzzy-Logic, Smart-Time-Fuzzy-Logic) in prototype Maquette. Each model will be discussed independently.

A. Maquette-Based Fixed-Time Model 1

It is a description of the current normal state of the roads that expresses the constant duration of the signal for the regular opening of each traffic light in order with a pre-established time limit for the opening of each signal, which has been set at 20 sec and here, open signal a, then count for 20 sec, then open signal b, then count for 20 sec, then open signal d, then count for 20 sec, and so on.

B. Maquette Based Smart-Time Model 2

It's a description of this system It takes into account the density first, as it opens the densest road and in the case of two vehicles, it opens the signal for 20 and 15 sec in the case of one vehicle. In all condition weather. This system can't open for null route from vehicle.

C. Maquette Based Smart-Time-Fuzzy-Logic Model 3

This system chosen will route depend on the highest number of vehicles in any route. The route has highest number of vehicles their route will be open first and so on. If there are two routes have the same number of cars the light open for the route that have less than time. The same route can't be open twice in the same turn even if it has the highest number of vehicles. If there is no vehicle in all routes the light is become red. Counter time is flexibility change depend on the number of vehicle and the rain value as shown in Algorithm 1.

D. Maquette Based Fixed-Time-Fuzzy-Logic Model 4

This system change sequentially that mean it start from route A until route D and repeat the turn again. If any route doesn't have vehicle the controller will be skipped to the next route and if there is no vehicle in all routes the light is become red. Counter time is flexibility change depend on the number of vehicle and the rain value.

Algorithm 1: Maquette Smart-Time-Fuzzy-Logic Traffic Light System

Data: Input data will be the IR Sensors and Rain sensor
bool: $iR_{a1}, iR_{a2}, iR_{b1}, iR_{b2}, iR_{c1}, iR_{c2}, iR_{d1}, iR_{d2}$
integer: rain Level
Result: Output data is the **LED** counter and the traffic **LEDs**
begin
  while true do
    if first run then
      jam Roads[4];
      jam Times[4];
      jam Values[4];
      jam A ← 0;
      jam B ← *0;*
      jam C ← *0;*
      jam D ← *0;*
      size ← *4;*
      counter ← 0;
      current Route ← *default;*

```
                    Change Route(current
            Route;
            end
            Fuzzy (Cars, Rain);
            Calculate Jam();
            Calculate Jam Values();
            Sort ();
            Display (counter);
            Change Route(current Route);
            counter ←
            counter - 1;
      endend
```

Fuzzy logic :

```
Function Fuzzy (){

      function input crisp Value (input){

            input=input fuzzy

      }
      function fuzzification (a, b, c, d) {

            if b equal to c AND c not equal to d
      is True than

                  if input < a || d < input
            then

                        member ship
                  Value = 0 ;
                  end

                  else if a <=input && b > =
                  input then

                        member ship
                  Value = (input-a) / (b-a);
                  end

                  else if a < input && input
                  > b then

                        member ship
                  Value = (input-c) / (d-c);
                  end
            end
            else if a not equal b AND b not
            equal c AND c not equal d True
            then

                  if input < a || d < input
            then

                    member ship Value = 0;
                  end

                  else if input > = b &&
                  input <= d && input <= c
                  then
                        member ship
                  value = 1;
                  end
                  else if a <= input && b
                  >= input then

                        member ship
                  Value = (input-a) / (b-a);
                  end
                  else if a < input && input
            > b && input < d

                        member ship
                  Value = (input-c) / (d-c);
                  end
            end
            else if c equal to d
                  if i input < a

                    member ship Value=0;
                  end
                  else if a < = input && b>=
input

                        member ship
                  Value = (input-a) / (b-a);
                  end
                  else if input >=b

                        member ship
                  value=1;
                  end
            end
      }

      Out Fuzzy Value (a, b ,c, d) {

            if b == c &&c != d
                  defuzzify centroid = b;
            end
            else if (a !=b && b != c && c != d)
                  defuzzify centroid = (b + c
            ) /2;
            end
            else if ( c == d )

                  defuzzify centroid = (b + c)
            /2;
            end
      }
      defuzzify (input1, input2, defuzzifycentroid,
operator ) {

            if Operator == AND
                  Aggregation result = min
            (input1, input2);
                  Crisp_output = defuzzify
            centroid * Aggregation result;
            end
            else if Operator == OR
```

```
                    Aggregation result =
            max(input1,input2);

                    Crisp _output = defuzzify
            centroid * Aggregation result;
            end
    }}
```

Calculatejam Function :

```
Function Calculatejam ():
    if iRa1 is true and iRa2 is true then
            jamA← 2;
    else iRa1 is true or iRa2 is true then
            jamA← 1;
    else
            jamA← 0;
    end
    if iRb1 is true and iRb2 is true then
            jamB← 2;
    else iRb1 is true or iRb2 is true then
            jamB← 1;
    else
            jamB ← 0;
    end
    If iRc1 is true and iRc2 is true then
            jam C← 2;
    else iRc1 is true or iRc2 is true then
            jam C← 1;
    else
            jam C ← 0;
    end
    if iRd1 is true and iRd2 is true then
            jamD ← 2;
    else iRd1 is true or iRd2 is true then
            jamD← 1;
    else
            jamD← 0;
    end
```

Calculate Jam Values Function :

```
Function calculate Jam Values ():
    if jamA is 0 then
        if route A is In Array (jam
    Roads)then
            ind A ← Get Index (route
        A);
            unset Jam Roads[ind A];
            unset Jam Times[ind A];
            unset jam Values[indA];
        end
    else
        if route A is In Array (jam Roads)
    and Jam A is changed then
            ind A ← Get Index (route
        A);
            Jam Times [ind A]←
        NOW;
            jam Values[ind A]←jam
        A;
        end
        else
            Jam Roads. add(route A);
            Jam Times. add(NOW);
            jam Values. add(jam A);
        end
    end
    if jam B is 0 then
        if route B is In Array (jam
    Roads)then
            ind B ← Get Index (route
        B);
            unset Jam Roads[ind B];
            unset Jam Times[ind B];
            unset jam Values[ind B];
        end
    else
        if route B is In Array (jam Roads)
    and Jam B is changed then
            ind B ← Get Index (route
        B);
            Jam Times[ind B]←NOW;
            Jam Values[ind B]←jam
        B;
        end
        else
            Jam Roads .add(route B);
            Jam Times. add (NOW);
            Jam Values. add(jam B);
        end
    end

    if jam C is 0 then
        if route C is In Array (jam
    Roads)then
            ind C ← Get Index (route
        C);
            unset Jam Roads[ind C];
            unset Jam Times[ind C];
            unset jam Values[ind C];
        end
    else
        if route C is In Array (jam
    Roads)and Jam C is changed then
            ind C ←Get Index (route
        C);
            Jam Times  [ind C]←
        NOW;
            Jam Values[ind C]←jam
        C;
        end
        else
            Jam Roads. add(route C);
            Jam Times. add(NOW);
            Jam Values. add(jam C);
        end
    end
    if jamD is 0then
        if route D is In Array (jam
    Roads)then
            ind D ← Get Index (route
        D);
            unset Jam Roads[ind D];
            unset Jam Times[ind D];
```

```
                    unset jam Values[ind D];
                end
        else
                if route D is In Array (jam
        Roads)and Jam D is changed then
                    ind D ← Get Index (route
                D);
                    Jam Times[ind D]←NOW;
                    Jam Values[ind D]←jam
                D;
                end
                else
                    Jam Roads. add(route D);
                    Jam Times. add(NOW);
                    Jam Values. add(jam D);
                end
        end
end
```

Sort Function:

```
Function Sort ():
        for i = 0<size do
                forj = i + 1 < size do
                    if jam Values[i]< jam
                Values[j]then
                            tmp←jam
                        Values[i];
                            jam Values[i]←
                        jam Values(j);
                            jam
                        Values[j]←tmp;
                            tmp char ← jam
                        Roads[i];
                            jam Roads[i]←
                        jam Roads(j);
                            jam
                        Roads[j]←tmp char;
                            tmp←jam
                        Times[i];
                            jam Times[i]←
                        jam Times[j];
                            jam
                        Times[j]←tmp;
                        end; end; end;
        fori=0 < size do
                forj = i + 1 < size do
                if jam Values[i] ==jam Values[j]an
                Times[i]> jam Times[j]then
                            tmp← jam
                        Values[i];
                            jam Values[i]←
                        jam Values[j];
                            jam
                        Values[j]←tmp;
                            tmp char  ←jam
                        Roads[i];
                            jam Roads[i]←
                        jam Roads[j];
                            jam
                        Roads[j]←tmp char;
                            tmp←jam
                        Times[i];
                            jam Times[i]←
                        jam Times[j];
                            jam
                        Times[j]←tmp;
                        end;end; end;
            max Jam ←jamValues[0];
```

Display Function :

```
Function display (counter):
Sevseg. set number (counter)
        if counter == 0 and (current Route ! = max
jam values [0]== 0 ) then
                if jam value [1] ! = 0 then
                        current Route ←jam
                Roads [1];
            else
                        current Route ←jam
                Roads [0];
                end
                        current Route ←get
                counter (jam Roads [1]) ;
                end
```

get Counter Function :

```
Function get Counter (cars Number):
        if NO Rain then
                if cars Number == 1then
                        return 15;
                elseif cars Number = 2 and cars
        Number < 5 then
                        return 20;
        end
        elseif Light Rain then
                if cars Number == 1then
                        return 15;
                elseif cars Number =2 and cars
        Number < 5 then
                        return 22;
        end
        elseif Normal Rain then
                if cars Number == 1then
                        return 15;
                elseif cars Number = 2 and cars
        Number <5 then
                        return 26;
        end
        elseif High Rain then
                if cars Number == 1then
                        return 15;
                else if cars Number = 2 and cars
        Number < 5then
                        return 30;
         end
```

Simulation-Based Traffic Light System

This Section expands experiments with more than 100 in 10 min vehicles distributed randomly with different air conditions.

Simulation-Based Fixed-Time Model 5

As mentioned before, this system opens traffic lights in ordered manner regardless of the road density. The control system opens the light for 20 sec in all weather conditions. Each vehicle can take 5, 7, 9, and 11 sec to exit from the road based on weather condition.

B. Simulation-Based Smart-Time Model 6

It's a description of this system It takes into account the density first, as it opens the densest road and in the case of one vehicle, it opens the signal for 15 or 20 sec in the case of two vehicles and 35 sec in the case of five vehicles in all condition weather. The one-only vehicle can take 5, 7, 9, and 11 sec to pass from route based on weather condition. This system can't open for null route from vehicle.

C. Simulation Based Fixed-Time-Fuzzy-Logic Model 7

This system change sequentially that mean it start from route A until route D and repeat the turn again. If any route doesn't have vehicle the controller will be skipped to the next route and if there is no vehicle in all routes the light is become red. Counter time is flexibility change depend on the number of vehicle and the rain value.

D. Simulation Based Smart-Time-Fuzzy-Logic Model 8

This system chosen will route depend on the highest number of vehicles in any route. The route has highest number of vehicles their route will be open first and so on. If there are two routes have the same number of cars the light open for the route that have less than time. The same route can't be open twice in the same turn even if it has the highest number of vehicles. If there is no vehicle in all routes the light is become red. Counter time is flexibility change depend on the number of vehicle and the rain value. As shown in Algorithm 2, the pseudocode of the proposed simulated smart time fuzzy logic system is displayed.

Algorithm 2: Simulated-Smart-Time-Fuzzy-Logic-Traffic Light System

```
Data: Input Data for Smart Simulation System
integer: rain Level, car Numbers, total System Time
Result: Output data is the LED counter and the traffic LEDs
begin
        for i=0; i< car Numbers; i ++ do
                random Entering Time[i]←random Time();
                random Route[i] Random Route();
        end
        while true do
                if first run then
                jam Roads[4];
jam Times[4];
                jam Values[4];
                        jam A ←0;
                        jam B ←0;
                        jam C ←0:
                        jam D ←0;
                        size ←4;
                        counter ←0;
                        current Route ←default;
                        Change Route (current Route);
                end
                simulation();
                Fuzzy (Cars, Rain);
                Calculate jam Values();
                Sort ();
                Display (counter);
                Change Route (current Route);
        counter ←
                counter-1;
        endend
```

Simulation Function:

```
Function simulation():
        for i = 0; i< car Numbers; i ++ do
                if random Entering Time[i] == now()then
                        if random Route[i] =='a' then
                                jam A++;
                        end
                        else if random Route[i] =='b' then
                                jam B ++;
                        end
                        else if random Route[i] =='c' then
                                jam C + + ;
                        end
                        else if random Route[i] =='d' then
                                jam D + + ;
                        end
                end
                if random Entering Time[i] + get Car Leaving Time()) == (now( )then
                        if random Route[i] =='a' then
                                jam A - -;
                        end
                        else if random Route[i] =='b' then
                                jam B - - ;
                        end
                        else if random Route[i] =='c' then
                                jam C - - ;
                        end
                        else if random Route[i] =='d' then
                                jam D - - ;
                        end; end;end;
```

Fuzzy logic:

```
Function Fuzzy () {
        function input crisp Value (input){
                input=input fuzzy
        }
        function fuzzification (a, b, c, d) {
                if b equal to c AND c not equal to d is True than
                        if input < a || d < input then
                        member ship Value = 0 ;
                        end
                        else if a <=input && b > = input then
                                member ship Value = (input-a) / (b-a);
                        end
                        else if a < input && input > b then
                                member ship Value = (input-c) / (d-c);
                        end             end
                else if a not equal b AND b not equal c AND c not equal d True then
                        if input < a || d < input then
                          member ship Value = 0;
                        end
                        else if input > = b && input <= d && input <= c then
                                member ship value = 1;
                        end
                        else if a <= input && b >= input then
                                member ship Value = (input-a) / (b-a);
                        end
                        else if a < input && input >b && input < d
```

```
                                    member ship
                              Value = (input-c) / (d-c);
                              end       end
                    else if c equal to d

                         if i input < a

                           member ship Value=0;
                         end
                        else if a < = input && b>=
input

                                    member ship
                              Value = (input-a) / (b-a);
                              end
                              else if input >=b

                                    member ship
                              value=1;
                              end; end;
          }
          Out Fuzzy Value (a, b ,c, d) {
                    if b == c &&c != d
                              defuzzify centroid = b;
                    end
                    else if (a !=b && b != c && c != d)
                              defuzzify centroid = (b + c
                    ) /2;
                    end
                    else if ( c == d )
                              defuzzify centroid = (b + c)
                    /2;
                    end       }
          defuzzify (input1, input2, defuzzifycentroid,
operator ) {if Operator == AND
                              Aggregation result = min
                    (input1, input2);
                              Crisp_output = defuzzify
                    centroid * Aggregation result;
                    end
                    else if Operator == OR
                              Aggregation result =
                    max(input1,input2);
                              Crisp _output = defuzzify
                    centroid * Aggregation result;
                    end       }}
```

Calculate Jam Values Function :

```
Function calculate Jam Values ( ):
          if jam A is 0 then
                    if route A is In Array (jam
          Roads)then
                              ind A ← Get Index (route
                    A);
                              unset Jam Roads[ind A];
                              unset Jam Times[ind A];
                              unset jam Values[indA];
                    end
          else
                    if route A is In Array (jam Roads)
          and Jam A is changed then
                              ind A ← Get Index (route
                    A);
                              Jam Times [ind A] ←
                    NOW;
                              jam Values[ind A] ← jam
                    A;
                    end
                    else
                              Jam Roads. add(route A);
                              Jam Times. add(NOW);
                              jam Values. add(jam A);
                    end
          end
          if jam B is 0 then
                    if route B is In Array (jam
          Roads)then
                              ind B ← Get Index (route
                    B);
                              unset Jam Roads[ind B];
                              unset Jam Times[ind B];
                              unset jam Values[ind B];
                    end
          else
                    if route B is In Array (jam Roads)
          and Jam B is changed then
                              ind B ← Get Index (route
                    B);
                              Jam Times[ind B] ← NOW;
                              Jam Values[ind B] ← jam
                    B;
                    end
                    else
                              Jam Roads .add(route B);
                              Jam Times. add (NOW);
                              Jam Values. add(jam B);
                    end;
          end
          if jam C is 0 then
                    if route C is In Array (jam
          Roads)then
                              ind C ← Get Index (route
                    C);
                              unset Jam Roads[ind C];
                              unset Jam Times[ind C];
                              unset jam Values[ind C];
                    end
          else
                    if route C is In Array (jam
          Roads)and Jam C is changed then
                              ind C ← Get Index (route
                    C);
                              Jam Times  [ind C] ←
                    NOW;
                              Jam Values[ind C] ← jam
                    C;
                    end
                    else
                              Jam Roads. add(route C);
                              Jam Times. add(NOW);
                              Jam Values. add(jam C);
                    Endend
```

```
        if jamD is 0then
                if route D is In Array (jam
        Roads)then
                        ind D ← Get Index (route
                D);
                        unset Jam Roads[ind D];
                        unset Jam Times[ind D];
                        unset jam Values[ind D];
                end
        else
                if route D is In Array (jam
        Roads)and Jam D is changed then
                ind D ← Get Index (route D);
                Jam Times[ind D]←NOW;
                Jam Values[ind D]←jam D;
                end
                else
                        Jam Roads. add(route D);
                        Jam Times. add(NOW);
                        Jam Values. add(jam D);
                End;end;end
```

Sort Function *:*

```
Function Sort ():
        for i = 0<size do
                for j = i + 1 < size do
                        if jam Values[i]< jam
                Values[j]then
                                tmp←jam
                        Values[i];
                                jam Values[i]←
                        jam Values(j);
                                jam
                        Values[j]←tmp;
                                tmp char ← jam
                        Roads[i];
                                jam Roads[i]←
                        jam Roads(j);
                                jam
                        Roads[j]←tmp char;
                                tmp←jam
                        Times[i];
                                jam Times[i]←
                        jam Times[j];
                                jam
                        Times[j]←tmp;
                        endendend
        fori=0 < size do
                for j = i + 1 < size do
                if jam Values[i] ==jam Values[j]an
                Times[i]> jam Times[j]then
                                tmp← jam
                        Values[i];
                                jam Values[i]←
                        jam Values[j];
                                jam
                        Values[j]←tmp;
                                tmp char ←jam
                        Roads[i];
                                jam Roads[i]←
                        jam Roads[j];
                                jam
                        Roads[j]←tmp char;
                                tmp←jam
                        Times[i];
                                jam Times[i]←
                        jam Times[j];
                                jam
                        Times[j]←tmp;
                        endendend
        max Jam ←jamValues[0];
```

Display Function *:*

```
Function display(counter):
Sevseg. set number (counter)
        if counter == 0 and (current Route ! = max
jam values [0]== 0 ) then
                if jam value [1] ! = 0 then
                        current Route ←jam
                Roads [1];          else
                        current Route ←jam
                Roads [0]; end
                        current Route ←get
                counter (jam Roads [1]) ;
                end
```

get Counter Function:

```
Function get Counter (cars Number):
        if NO Rain then
                if cars Number == 1then
                        return 15;
                elseif cars Number >= 2 and cars
        Number < 5 then
                        return 20;
                else
                        return 35;
                endend
        elseif Light Rain then
                if cars Number == 1then
                        return 21;
                elseif cars Number >=2 and cars
        Number < 5 then
                        return 28;
                else
                        return 49;
                end
        end
        elseif Normal Rain then
                if cars Number == 1then
                        return 27;
                elseif cars Number >= 2 and cars
        Number <5 then
                        return 36;
                else
                        return 63;
                end
        end
        elseif High Rain then
                if cars Number == 1then
                        return 33;
                else if cars Number >= 2 and cars
        Number < 5then
                        return 44;
```

```
            else
                    return 66;
            end; end;
get Car Leaving Time Function:
Function get Car Leaving Time( ):
        if NO Rain then
            return 5;
        end
        elseif Light Rain then
            return 7;
        end
        elseif Normal Rain then
            return 9;
        end
        else if High Rain then
            return 11;
        end
```

Rule Base of the Proposed Models

This section presents all rules of eight scenarios with real World Maquette and Simulation Software.

Maquette-Based Traffic Light System

A. Maquette Based Fixed-Time Model 1

- This system always opens a signal to each road arranged if found vehicle or not
- All signals always open 20 sec for each road if enter one vehicle or more in the road in all condition rain

B. Maquette Based Smart-Time Model 2

- All signals always open 15 in case of one vehicle or 20 sec in case of two vehicle for any road in all condition rain
- This system based on density road first and then less density road and so on
- This system can't open any signal for any road if no vehicle entered any road
- All signal is red if sensor no detect any vehicle in any road

C. Maquette Based Fixed-Time-Fuzzy-Logic Model 3

This system opens roads regularly with variable green time based on density of vehicle and rain condition but not open a road when it has no vehicles as shown:

- All signals open 15 or 20 sec for any road in case without rain
- All signals open 15 or 22 sec for any road in case with light rain
- All signals open 15 or 26 sec for any road in case with normal rain
- All signals open 15 or 30 sec for any road in case with high rain
- All signal is red if sensor no detect any vehicle in any road

D. Maquette Based Smart-Time-Fuzzy-Logic Model 4

This system is based on density road first and then less density road and so on with variable green time based on density and rain conditions as shown:

- All signals open for 15 or 20 sec for any road in the case without rain
- All signals open for 15 or 22 sec for any road in the case of light rain
- All signals open for 15 or 26 sec for any road in the case of normal rain
- All signals open for 15 or 30 sec for any road in the case of high rain
- All signal is red if the sensor no detect any vehicle on any road

Simulation-Based Traffic Light System

A. Simulation-based Fixed-Time Model 5

- This System always open a signal to each road arranged if found vehicles or not
- All signals always open 20 sec for each road if enter one vehicle or more in the road in all condition rain

B. Simulation-Based Smart-Time Model 6

- All signals always open 15 sec for each road if enter one vehicle in the road in all condition rain
- All signals always open 20 sec for each road if enter two vehicles in the road in all condition rain
- All signals always open 35 sec for each road if enter five vehicles in the road in all condition rain
- This system based on density road first and then less density road and so on
- This system can't open any signal for any road if no vehicle entered any road
- All signal is red if sensor no detect any vehicle in any road

C. Simulation Based Fixed-Time-Fuzzy-Logic Model 7

This system opens roads regularly with variable green time based on density vehicle and rain conditions but does not open a road when it has no vehicles as shown:

- All signals can be open 15 or 20 or 35 sec on any road in case without rain
- All signals open 21 or 28 or 49 sec any road in case with light rain
- All signals open 27 or 36 or 63 sec for any in case with normal rain
- All signals open 33 or 44 or 66 sec for any road in case with high rain
- All signal is red if sensor no detect any vehicle in any road

D. Simulation-Based Smart-Time-Fuzzy-Logic Model 8

This system is based on density road first and then less density road and so on with variable green time based on density and rain conditions as shown:

- All signals can be open 15 or 20 or 35 sec for on any road in case without rain
- All signals open 21 or 28 or 49 sec for any road in case with light rain
- All signals open 27 or 36 or 63 sec for any in case with normal rain
- All signals open 33 or 44 or 66 sec for any road in case with high rain
- All signal is red if sensor no detect any vehicle in any road

Experimental Results and Discussion

Performance Evaluations

Three evaluation metrics are used to evaluate the experimental results.

A. Waiting Time of Vehicle

To estimate the car waiting time, the following equation is used:

$$W_i = T_{i2} - T_{i1} \quad (1)$$

where, W is the waiting time, (i) is the car number, T_{i2} is the time after the car left the road and T_{i1} is the time when the car entered the road.

B. Total Waiting Time

Equation 2 is used to calculate the total waiting time of car:

$$T_c = \sum_{i=0}^{n} W_i \quad (2)$$

where, W is the waiting time, (i) is the car number and n is the number of cars.

C. Average Waiting Time

The average waiting time of car is calculated by:

$$A_c = \frac{T_c}{n} \quad (3)$$

where, T_c is the total waiting time and n is the number of cars.

Experimental Results

As stated above, the proposed prediction algorithms are tested in two separate methods: Real Maquette with low congestion and simulation software with high congestion. All of them tested with four states of rain (no-rain, low-rain, medium-rain, high-rain) and low congestion, and high congestion.

A. Results of Maquette Real Traffic Signal

Table 1 shows the results of the average leaving waiting time of the proposed four approaches in different four weather condition with 60 Road Movement cases. While Table 2 shows the improvement percentages and compares the results with fixed traditional time in Light-Rain, medium-Rain, and Heavy-Rain condition consequently.

After comparing the results of the previous four systems, it is possible to occur on the roads which simulates reality by using a four-way micro simulator model in which each method can contain two vehicles. The best of the previous systems was found to be a fuzzy logic system both for fixed and smart fuzzy logic.

Results of Simulation Traffic Signal

This section will discuss the proposed models executed on the simulation platform. Table 3 shows the results of the proposed four approaches in normal weather (No- Rain) in terms of average waiting time and total waiting time. While Table 4 compares the results in Low-Rain condition, Table 5 in case of medium-rain Condition, and Table 6 with the High rain Condition. A simulation system has been done on both smart-time systems, smart-time-fuzzy-logic system, fixed-time fuzzy-system, and fixed-time-system, but with a time limit, and here are 10 min that can be changed by deficiency or increase. Also specified are the number of vehicles that will pass through the roads during this time. Here are 100 vehicles that can be changed by deficiency or increase as the waiting time for each vehicle appears until it passes through the road or path that will also appear. Finally, total Waiting Time for all vehicle, average waiting time for each vehicle, and the total time taken to pass through all vehicles are calculated. All results are through the smart time system, fixed time system, fuzzy logic fixed time system, and smart fuzzy logic system programs. These results will also be calculated by replicating the simulator (3) times each and the results and explanations are as follows.

Discussion

Eight different models have been thoroughly researched and it has been found that the best model is that which is based on the smart fuzzy logic system. The proposed fuzzy model enhanced the total time for all vehicles from 2490 to 1154 sec the average waiting time of the vehicles also reduced from 769 to 289 sec in heavy rain condition resulting in 2.7 times reduction in processing speed. The proposed model contributed to many solutions such as: Reduction of waiting time which leads to a reduction in fuel consumption and air pollution.

Table 1: Results of leaving average waiting time of the proposed four maquette models

Case	Fixed time	Smart time	Fixed fuzzy	Smart fuzzy
Normal state	65	42	42	42
Low rain	146	75	47	47
Normal rain	145	74	55	55
High rain	145	74	64	64

Table 2: The improvement percentage comparison of the smart and fuzzy systems with the fixed system by a maquette

Case	Fixed traditional time	Smart time	Fixed fuzzy	Smart fuzzy
Normal state	0%	35.38%	35.38%	35.38%
Low rain	0%	48.63%	67.80%	67.80%
Normal rain	0%	48.96%	62.06%	62.06%
High rain	0%	48.96%	55.86%	55.86%

Table 3: Results of Simulation-based traffic light System in (No-rain) Condition

No.	Time metric	Fixed time	Smart time	Smart fuzzy logic	Fixed fuzzy logic
1	Average time	57s	45s	38s	45s
	Total time	725s	693s	632s	655s
2	Average time	79s	53s	37s	41s
	Total time	814s	694s	645s	67s
3	Average time	55	42	37	41
	Total time	720	681	637	639

Table 4: Results of simulation-based traffic light system in (Light-rain) condition

No.	Time metric	Fixed time	Smart time	Smart Fuzzy logic	Fixed fuzzy logic
1	Average time	286s	142s	120s	125s
	Total time	1485s	827s	784s	784s
2	Average time	299s	137s	110s	122s
	Total time	1296s	802s	745s	799s
3	Average time	249s	144s	106s	125s
	Total time	1184s	859s	784s	784s

Table 5: Results of simulation-based traffic light System in (Medium-rain) condition

No.	Time metric	Fixed time	Smart time	Smart fuzzy logic	Fixed fuzzy logic
1	Average time	289s	286s	194s	202s
	Total time	1489s	1205s	967s	974s
2	Average time	300s	288s	185s	234s
	Total time	1258s	1184s	956s	1009s
3	Average time	302s	295s	206s	212s
	Total time	1300s	1178s	944s	1027s

Table 6: Results of simulation-based traffic light System in (Heavy-rain) condition

No.	Time metric	Fixed time	Smart time	Smart fuzzy logic	Fixed fuzzy logic
1	Average time	769s	389s	289s	312s
	Total time	2490s	1350s	1154s	1162s
2	Average time	822s	364s	309s	334s
	Total time	2532s	1333s	1130s	1198s
3	Average time	754s	383s	268s	313s
	Total time	2175s	1315s	1156s	1199s

Conclusion

In densely populated areas, several intersections are linked. In Egypt, these crossings can sometimes lead to endless traffic density and long queues of traffic. The idea behind smart traffic systems is that drivers will not spend unnecessary time waiting for traffic lights to change. In this study, eight scenarios are simulated using the Arduino IDE and real Maquette to help address this type of traffic challenge. The acquired road-related data such as (density and rain conditions) are used in the fuzzy model to optimize the time of green light. The main goal of the proposed smart traffic light system is to reduce the waiting time for each vehicle's lane and to maximize the total number of cars that can cross the intersection. To solve the above problems, we used a smart fuzzy logic control system' algorithm to detect

the density for every road and select the most stable time for the cars in every road with consideration of the rain condition if any. The IR sensor can be used to detect the density and the rain sensor to define the road condition. The proposed fuzzy model enhanced the total time for all vehicles from 2490 to 1154 sec The average waiting time of the vehicles also reduced from 769 to 289 sec in heavy rain condition. The proposed model contributed to many solutions such as: Reduction of waiting time which leads to a reduction in fuel consumption and air pollution. Furthermore, Favoritism is eradicated as the system strictly follows the programmed sequence regardless of the personnel in the queue. The mistake of assigning green light time to a particular route over and over again to the detriment of others can be overcome. There is no fixed time for a route, the green light time is flexible depending on traffic density and weather conditions. The system can work 24 h depending on the power supply. The system provided increased rate of traffic and more stability. In future work, new optimization techniques such as Genetic algorithm and particle swarm optimization will be used to optimize the parameters of the smart traffic light system.

Acknowledgment

This study was supported by The Information Systems Department, Faculty of Computers & Information, Fayoum University, Fayoum, Egypt.

Funding Information

This research wasn't funded by any organization. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author's Contributions

Adel Abdallah Abdou: Developed the implementation and the system design. Also, he has conducted the literature review and performed the analysis. He provided the first paper draft and interpretated the results.

Mohamed Hassan Farrag and A. S. Tolba: Developed the methodology and performed the computations. He has supervised all the work and validated the idea.

Ethics

This manuscript substance is the author's original work and has not been previously published somewhere else. Authors already read and approved the manuscript and no potential ethical issues are immersed.

References

Adewale, A. L., Jumoke, A. F., Adegboye, M., & Ismail, A. (2018). An embedded fuzzy logic-based application for density traffic control system. *International Journal of Artificial Intelligence Research*, *2*(1), 7-16. http://ijair.id/index.php/ijair/article/view/44

Adoni, W. Y. H., Aoun, N. B., Nahhal, T., Krichen, M., Alzahrani, M. Y. & Mutombo, F. K. (2022). A Scalable Big Data Framework for Real-Time Traffic Monitoring System. Journal of Computer Science, 18(9), 801-810. https://www.researchsquare.com/article/rs-1200646/latest.pdf

Agarwal, H., & Rai, J. N. (2022, March). Traffic Control System based on Density with Emergency Priority Mechanism. In *2022 International Conference on Electronics and Renewable Systems (ICEARS)* (pp. 192-197). IEEE. https://ieeexplore.ieee.org/abstract/document/9751869

Ahmad, A., Arshad, R., Mahmud, S. A., Khan, G. M., & Al-Raweshidy, H. S. (2014). Earliest-deadline-based scheduling to reduce urban traffic congestion. *IEEE Transactions on Intelligent Transportation Systems*, *15*(4), 1510-1526. https://ieeexplore.ieee.org/abstract/document/6736123

Elda, M., Kuka, S., & Zhifka, M. (2019). Application of Fuzzy Logic in Traffic System. *International Journal of Innovative Research in Engineering & Management* (IJIREM), 6, 4. https://doi.org/10.21276/ijirem.2019.6.4.2

Hlavacek, J., & Gotthans, J. (2022, April). Evaluation of Traffic Crossings Situations using Machine Learning. In *2022 32nd International Conference Radioelektronika (RADIOELEKTRONIKA)* (pp. 01-04). IEEE. https://ieeexplore.ieee.org/abstract/document/9764937/

Kammoun, H. M., Kallel, I., Casillas, J., Abraham, A., & Alimi, A. M. (2014). Adapt-Traf: An adaptive multiagent road traffic management system based on a hybrid ant-hierarchical fuzzy model. *Transportation Research Part C: Emerging Technologies*, *42*, 147-167. https://doi.org/10.1016/j.trc.2014.03.003

Mualifah, L. N. A., & Abadi, A. M. (2019, October). Optimizing the traffic control system of Sultan Agung street Yogyakarta using fuzzy logic controller. In *Journal of Physics: Conference Series* (Vol. 1320, No. 1, p. 012028). IOP Publishing. https://iopscience.iop.org/article/10.1088/1742-6596/1320/1/012028/meta

Nigam, A., Chaturvedi, M., & Srivastava, S. (2022, January). An Empirical Study on Parameters Affecting Traffic Stream Variables Under Rainy Conditions. In *2022 14th International Conference on COMmunication Systems & NET work S (COMSNETS)* (pp. 818-823). IEEE. https://ieeexplore.ieee.org/abstract/document/9668377

Patel, D., & Rohilla, Y. (2020, September). Infrared sensor based self–adaptive traffic signal system using arduino board. In 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 175-181). IEEE. https://ieeexplore.ieee.org/abstract/document/9242560

Studyana, M. D., Sjafruddin, A., Kusumantoro, I. P., & Soeharyadi, Y. (2018, October). Fuzzy logic model for traffic signal intersection. In AIP Conference Proceedings (Vol. 2021, No. 1, p. 060020). *AIP Publishing LLC*. https://doi.org/10.1063/1.5062784

Tomar, I., Indu, S., & Pandey, N. (2022). Traffic Signal Control Methods: Current Status, Challenges, and Emerging Trends. *Proceedings of Data Analytics and Management*, 151-163. https://doi.org/10.1007/978-981-16-6289-8_14

Xiao, F. (2022). CaFtR: A fuzzy complex event processing method. *International Journal of Fuzzy Systems*, *24*(2), 1098-1111. https://doi.org/10.1007/s40815-021-01118-6

Yang, Z., Huang, W., & Chen, X. (2022). Mitigation of Rain Effect on Wave Height Measurement Using X-Band Radar Sensor. *IEEE Sensors Journal*, *22*(6), 5929-5938. https://ieeexplore.ieee.org/abstract/document/9706435