**Introducere**

**Logica *fuzzy***  a fost definită în 1965 de către prof. Lotfi Zadeh de la Universitatea California din Berkeley. Spre deosebire de logica clasică (binară), în care se lucrează cu două valori de adevăr exacte (notate de ex. 0 pentru fals și 1 pentru adevărat), logica*fuzzy* folosește o plajă continuă de valori logice cuprinse în intervalul [0, 1], unde 0 indică falsitatea completă, iar 1 indică adevărul complet. Astfel, dacă în logica clasică un obiect poate aparține (1) sau nu (0) unei mulțimi date, logica  *fuzzy* lucrează cu gradul de apartenență al obiectului la mulţime, acesta putând lua valori între 0 și 1.

Logica *fuzzy* oferă instrumentele necesare pentru reprezentarea în sistemele inteligente a unor concepte imprecise cum sunt „mare”, „fierbinte”, „ieftin” ș.a., concepte numite variabile lingvistice sau variabile  *fuzzy*. Pentru reprezentarea acestora se folosesc seturile *fuzzy*, care captează din punct de vedere cantitativ interpretarea calitativă a termenilor.

Bazate pe logica *fuzzy*, **sistemele *fuzz****y* sunt considerate un caz particular al sitemelor expert (motiv pentru care mai sunt denumite și sisteme expert *fuzzy*), sisteme care oferă o metodă flexibilă pentru tratarea incertitudinii.

Japonia este țara cu cele mai multe sisteme *fuzzy* implementate, mai ales în domeniile urmăririi producției și a vânzărilor. De asemenea, multe sisteme *fuzzy* au fost înglobate în unele dintre bunurile de larg consum: mașini de spălat rufe, cuptoare cu microunde, aparate foto, aparate de aer condiționat, etc.

Un sistem de control fuzzy utilizează o colecţie de funcţii de apartenenţă şi reguli pentru a raţiona pe baza cunoştinţelor date. Valorile crisp care reprezintă intrările unui astfel de sistem sunt transformate în valori fuzzy pentru a putea fi folosite în aplicarea regulilor sistemului, reguli ce sunt formulate prin intermediul expresiilorlingvistice. Cele mai multe sisteme fuzzy transformă şi concluzia lingvistică obţinută în valoare crisp.



Fig.1 Arhitectura unui controler fuzzy

Aşa cum se observă în figura de mai sus componentele unui controler fuzzy sunt:

***modulul de fuzzificare*** care transformă valorile de intrare crisp în mulţimi fuzzy

pentru a putea calcula valoarea de adevăr a premisei fiecărei reguli din baza de reguli pentru intrarea dată. În general intrările unui sistem fuzzy sunt valori crisp. În multe aplicaţii practice se normalizează domeniul intrărilor folosindu-se următorul tip de partiţie: NB (negativ mare), NM (negativ mediu), NS (negativ mic), ZE(zero), PS(pozitiv mic), PM (pozitiv mediu), PB (pozitiv mare).

***modulul de inferenţă***calculează valoarea de adevăr a premiselor din baza de reguli înraport cu datele de intrare şi aplică valorile obţinute concluziilor corespunzătoare obţinându-se astfel câte o mulţime fuzzy de ieşire pentru fiecare regulă.

***modulul de agregare***combină toate mulţimile fuzzy asignate concluziilor de către modulul de inferenţă pentru a forma o singură mulţime fuzzy.

***modulul de defuzzificare***are rolul de a converti mulţimea fuzzy obţinută de modululde agregare la o valoare crisp care va fi valoarea de ieşire a sistemului pentru intrarea dată.

**Referinte Stiintifice**

Pentru realizarea proiectului am consultat mai multe referinte stiintifice, dupa cum urmeaza:

**Extras din lucrarea *“Intelligent traffic lights control by fuzzy”* –** Tan Kok Khiang, Marzuki Khalid si Rubiyah Yusof -> Centrul pentru Inteligenta Artificiala si Robotica, Universitatea Tehnologica Malaysia.

Una dintre metodele de depasire a problemelor intampinate in trafic in marile orase consta in dezvoltarea unui sistem inteligent de monitorizare si control al semafoarelor. Aceasta lucrare prezinta proiectarea si implementarea unui astfel de sistem utilizand logica fuzzy.Astfel se propune inlocuirea semafoarelor conventionale cu timp fix cu semafoare cu timp variabil controlate prin logica fuzzy, beneficiind de proprietatea acestei logici de a cuantiza termeni si conditii ca de exemplu: mare, mic, etc.

Lucrarea de fata propune utilizarea unor senzori de proximitate care sa opereze o schimbare a culorii semaforului in momentul in care sunt prezente masini. O structura generala a sistemului e prezentata in figura urmatoare:

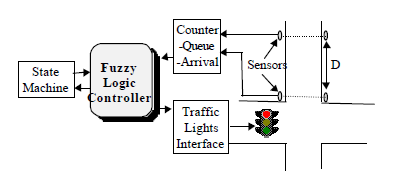


Fig.2 O structura generala a sistemului fuzzy de control a semaforului

Parametrii de intrare pentru modulul de control fuzzy sunt: marimea (dimensiunea) traficului pe axa in miscare si marimea (dimensiunea) traficului pe axa aflata in asteptare (ce va forma o coada). De exemplu daca semaforul de pe axa verticala (nord-sud) indica culoarea verde aceasta axa va fi cea in miscare, in acelasi timp axa orizontala (vest-est) va fi axa in asteptare, si vice-versa. Iesirea sistemului este reprezentata de extensia timpului cat lumina verde va fi activa pe axa in miscare.In functie de conditiile de trafic aceasta durata poate fi extinsa sau nu.

Exemple de reguli utilizate in proiect:

* **Daca** traficul de pe axa verticala este*aglomerat* **si** traficul pe axa orizontala este *lejer,* **atunci** permite miscarea pe axa verticala pentru *mai mult* timp.
* **Daca** traficul de pe axa verticala este *normal* **si** traficul pe axa orizontala este *normal,* **atunci**  miscarea se va desfasura *normal*.

In urma simularii comparative, in aceleasi conditii a sistemului bazat pe logica fuzzy creat cu un sistem bazat pe controlul semafoarelor cu timpi ficsi, s-a concluzionat ca sistemul bazat pe logica fuzzy datorita flexibilitatii sale realizeaza fluidizarea traficului.

**Extras din lucrarea ” A laser scanner for detailed traffic data collection and traffic control” -** K. Fürstenberg, J. Hipp, A. Liebram - 7th World Congress on Intelligent Transport Systems

Lucrarea de fata prezinta un scanner-laser (IBEO LD Laser) capabil sa obtina date detaliate statice si dinamice despre obiectele aflate la mare distanta. Acest dispozitiv are un unghi mare de deschidere (vizualizare) fiind potrivit pentru a aduna informatiile necesare. Principiul de functionare este asemanator radarelor, distanta fata de tinta fiind direct proportionala cu diferenta de timp dintre transmisia si receptia reflexiei aceluiasi impuls. Asigura o vizualizare a tuturor obiectelor aflate in jurul sau, putandu-se roti la 360° si avand o frecventa de scanare cuprinsa intre 10 si 40 de Hz.

O singura scanare poate monitoriza mai multe benzi simultan, in cazul in care strada este scanata in planul transversal. Astfel se pot extrage numarul obiectelor care trec printr-un punct la un moment dat. Daca scanner-ul este utilizat pentru a realiza o scanare pe un plan paralel cu strada se analizeaza vehiculele si se poate obtine o clasificare a acestora in functie de lungimea si inaltimea lor. Aceasta scanare ofera o clasificare detaliata si imparte participantii la trafic in: diferite tipuri de camioane, masini, motociclete, biciclete, etc., lucru exemplificat in figurile de mai jos:

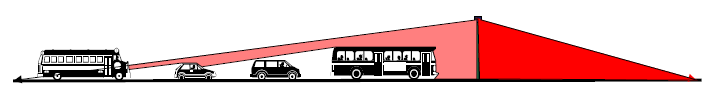


Fig.3 Scanare in plan paralel cu strada

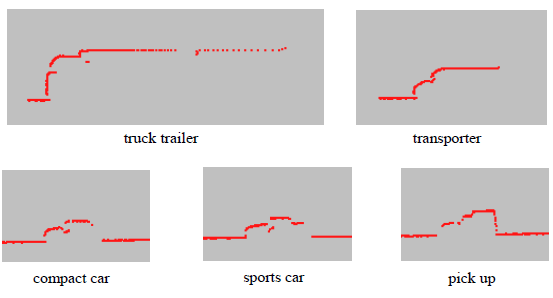


Fig.4 Date obtinute in urma scanarii diferitelor tipuri de vehicule

**Extras din lucrarea ”Intelligent traffic control and warning system and method”**– Jerome H. Lemelson, Robert D.Pedersen, Steven R. Pedersen – United States Patent (US 6633238 B2)

Lucrarea de fata prezintaun sistem de control al traficului si al semafoarelor si distribuirea de mesaje de avertizareselectiv pentru automobilisti. Sistemul include un controler pentru determinarea masurilor corespunzatoare in caz decongestionare a traficului si este realizat in logica fuzzy.

Este prezentat un sistem complex cu diferite marimi de intrare(de exemplu: viteza de deplasare, densitatea traficului) o serie de marimi de intrare auxiliare (de exemplu posibilitatea aparitiei unui accident in trafic), care genereaza la iesire atat modul in care vor fi controlate semafoarele, cat si mesaje de avertizare.

**Descrierea Sistemului Fuzzy utilizat**

Se doreste realizarea unui sistem inteligent de control al traficului utilizand logica fuzzy cu scopul de a preveni congestionarea traficului. Acest lucru se va realiza prin evaluarea unor parametri: densitatea traficului si viteza de deplasare a participantilor la trafic.

Sistemul fuzzy gandit are 2 intrari si o iesire:

* Intrarea 1 – densitatea traficului, exprimata in numar de vehicule / km;
* Intrarea 2 – viteza de deplasare, exprimata in numar de vehicule ce traverseaza intersectia / unitatea de timp (ora);
* Iesire – Numarul de secunde cu care va fi prelungita/scurtata perioada in care lumina verde a semaforului este active;

Fiecare din cele 2 intrari va fi reprezentata de 5 termeni lingvistici dupa cum urmeaza: “foarte mica (fm)”, “mica (m)”, “medie (med)”, “mare (M)”, “foarte mare (FM)”, iar fiecare termen lingvistic va fi aproximat prin 3 valori numerice. Reprezentarile grafice se gasesc in figurile de mai jos:

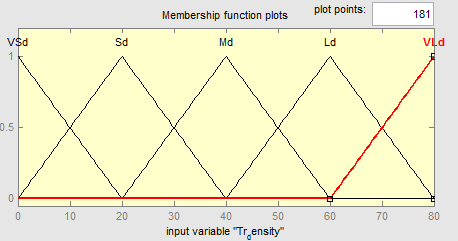


Fig. 5 Densitatea traficului (intrarea 1)

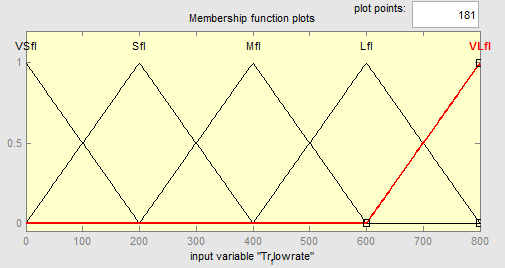


Fig. 6 Traffic flow rate (intrarea 2)

Iesirea va fi reprezentata de asemenea de 5 termeni lingvistici dupa cum urmeaza: “negativ mare (NM)”, “negativ mica (nm)”, “zero (z)”, “pozitiv mica (pm)”, “pozitiv mare (PM)”,fiecare termen lingvistic find aproximat prin 3 valori numerice. Reprezentarea grafica se gaseste in figura de mai jos:

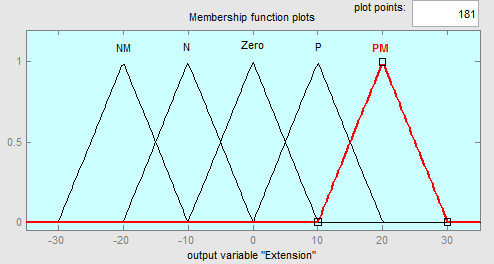


Fig. 7 Variabila de iesire (extensia nr. de secunde)

Gama de variatie a variabilei de iesire (extensia nr. de secunde) este cuprinsa intre -30 si 30 de secunde.

In functie de valorile variabilelor de intrare se realizeaza se realizeaza o tabela de reguli care va determina evolutia variabilei de iesire. Astfel pentru fiecare regulă se va obtine câte o mulţime fuzzy de ieşire.

Tabela de reguli este urmatoarea:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Flow rate\ Densitate | foarte mica (VSfl) | Mica (Sfl) | Medie (Mfl) | Mare (Lfl) | foarte mare (VLfl) |
| foarte mica (VSd) | Zero | P | PM | PM | PM |
| Mica (Sd) | N | Zero | P | PM | PM |
| Medie (Md) | NM | N | Zero | P | PM |
| Mare (Ld) | NM | NM | N | Zero | P |
| foarte mare (VLd) | NM | NM | NM | N | Zero |

1. IF *densitatea traficului* foarte mica (VSd) AND *flow rate* foarte mica (VSfl) THEN *extension* Zero;
2. IF *densitatea traficului* foarte mica (VSd) AND *flow rate* mica (Sfl) THEN *extension* negativ mica (N);
3. IF *densitatea traficului* foarte mica (VSd) AND *flow rate* medie (Mfl) THEN *extension* negativ mare (NM);
4. IF *densitatea traficului* foarte mica (VSd) AND *flow rate* mare (Lfl) THEN *extension* negativ mare (NM);
5. IF *densitatea traficului* foarte mica (VSd) AND *flow rate* foarte mare (VLfl) THEN *extension* negativ mare (NM);
6. IF *densitatea traficului* mica (Sd) AND *flow rate* foarte mica (VSfl) THEN *extension* pozitiv mica (P);
7. IF *densitatea traficului* mica (Sd) AND *flow rate* mica (Sfl) THEN *extension* Zero;
8. IF *densitatea traficului* mica (Sd) AND *flow rate* medie (Mfl) THEN *extension* negativ mica (N);
9. IF *densitatea traficului* mica (Sd) AND *flow rate* mare (Lfl) THEN *extension* negativ mare (NM);
10. IF *densitatea traficului* mica (Sd) AND *flow rate* foarte mare (VLfl) THEN *extension* negativ mare (NM);
11. IF *densitatea traficului* medie (Md) AND *flow rate* foarte mica (VSfl) THEN *extension* pozitiv mare (PM);
12. IF *densitatea traficului* medie (Md) AND *flow rate* mica (Sfl) THEN *extension* pozitiv mica (P);
13. IF *densitatea traficului* medie (Md) AND *flow rate* medie (Mfl) THEN *extension* Zero;
14. IF *densitatea traficului* medie (Md) AND *flow rate* mare (Lfl) THEN *extension* negativ mica (N);
15. IF *densitatea traficului* medie (Md) AND *flow rate* foarte mare (VLfl) THEN *extension* negativ mare (NM);
16. IF *densitatea traficului* mare (Ld) AND *flow rate* foarte mica (VSfl) THEN *extension* pozitiv mare (PM);
17. IF *densitatea traficului* mare (Ld) AND *flow rate* mica (Sfl) THEN *extension* pozitiv mare (PM);
18. IF *densitatea traficului* mare (Ld) AND *flow rate* medie (Mfl) THEN *extension* pozitiv mica (P);
19. IF *densitatea traficului* mare (Ld) AND *flow rate* mare (Lfl) THEN *extension* Zero;
20. IF *densitatea traficului* mare (Ld) AND *flow rate* foarte mare (VLfl) THEN *extension* negativ mica (N);
21. IF *densitatea traficului* foarte mare (VLd) AND *flow rate* foarte mica (VSfl) THEN *extension* pozitiv mare (PM);
22. IF *densitatea traficului* foarte mare (VLd) AND *flow rate* mica (Sfl) THEN *extension* pozitiv mare (PM);
23. IF *densitatea traficului* foarte mare (VLd) AND *flow rate* medie (Mfl) THEN *extension* pozitiv mare (PM);
24. IF *densitatea traficului* foarte mare (VLd) AND *flow rate* mare (Lfl) THEN *extension* pozitiv mica (P);
25. IF *densitatea traficului* foarte mare (VLd) AND *flow rate* foarte mare (VLfl) THEN *extension* Zero;

Graficul regulilor rezultat in urma utilizarii programului Matlab este:

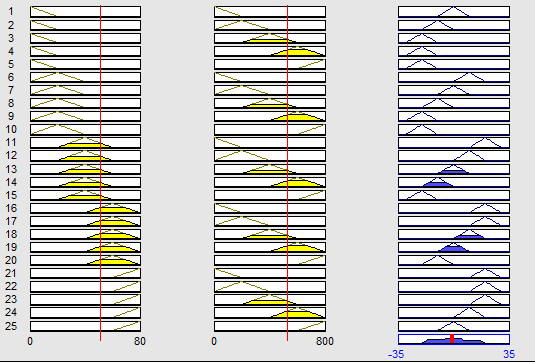


Fig. 8 Setul de reguli Fuzzy generat de Matlab

**Descrierea algoritmului**

Algoritmul a fost gandit pe baza urmatoarei organigrame:

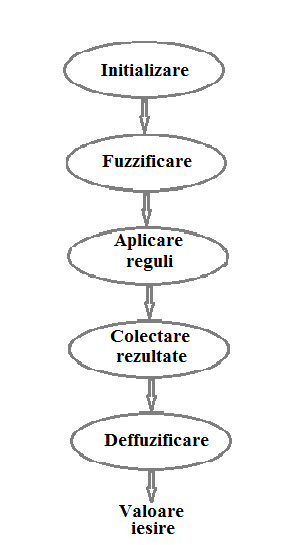


Fig. 9 Organigrama utilizata in descrierea sistemului

Sistemul Fuzzy functioneaza urmand cativa pasi mari:

1. Se initializeaza sistemul prin oferirea valorilor ‚crisp’ fiecarei variabile de intrare;
2. Se fuzzifica valorile furnizate ca intrari(transformarea valorile de intrare ‚’crisp’ în mulţimi fuzzy) pentru a putea calcula valoarea de adevăr a premisei fiecărei reguli din baza de reguli pentru intrarea dată.
3. Se calculează valoarea de adevăr a premiselor din baza de reguli înraport cu datele de intrare şi se aplică valorile obţinute concluziilor corespunzătoare obţinându-se astfel câte o mulţime fuzzy de ieşire pentru fiecare regulă.
4. Se combină toate mulţimile fuzzy asignate concluziilor de către modulul de inferenţă pentru a forma o singură mulţime fuzzy.
5. Etapa de defuzzificare - are rolul de a converti mulţimea fuzzy obţinută la o valoare crisp care va fi valoarea de ieşire a sistemului pentru intrarea dată.
6. Utilizarea (afisarea) valorii de iesire a sistemului.

**Rezultate si simulari**

Urmand pasii prezentati anterior am realizat un cod C (prezent la anexe) iar pentru diferite valori de intrare la iesire s-au obtinut urmatoarele rezultate:

|  |  |  |  |
| --- | --- | --- | --- |
| Nr.crt | Valoare densitate trafic | Valoare flow rate | Valoare iesire (extensie) |
| 1 | 52 | 450 | 2.52 |
| 2 | 70 | 120 | 20 |
| 3 | 24 | 520 | -13.05 |
| 4 | 60 | 600 | 0 |
| 5 | 32 | 425 | -6.04 |
| 6 | 52 | 600 | -4.19 |
| 7 | 34 | 250 | 3.15 |
| 8 | 60 | 300 | 15 |
| 9 | 20 | 600 | -20 |
| 10 | 56 | 360 | 10 |

Suprafata rezultata in urma simularii in programul Matlab este:

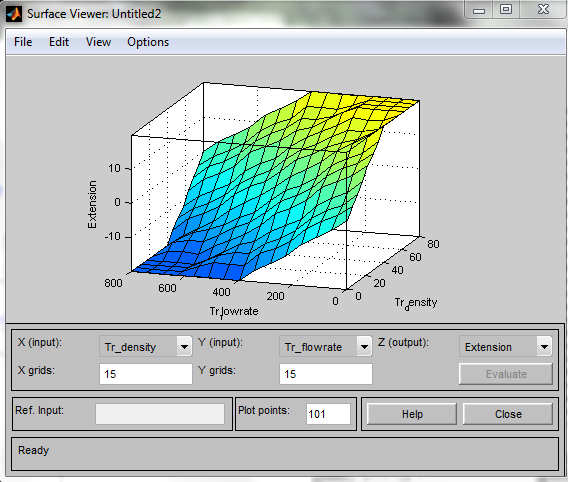


Fig. 10 Captura Matlab

Se poate observa ca valoarea variabilei de iesire poate varia intre -20 si 20 unitati de timp (secunde), fapt ce se poate observa si in tabelul de mai sus.

Pentru a arata ca sistemul este bine realizat am realizat o simulare in programul Matlab a primului set de valori din tabel. Rezultatul vorbeste de la sine.

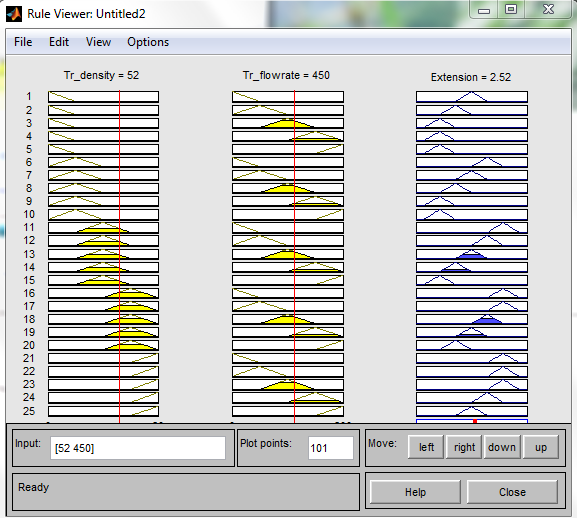


Fig.11 Rezultat Matlab

Pentru o mai buna intelegere a sistemelor Fuzzy am luat decizia de a studia cateva situatii care ar putea determina o schimbare a functionalitatii.

1. Primul studiu de caz ales este schimbarea tipului de sistem Fuzzy din Mamdani in Sugeno. In locul functiilor de iesire triunghiulare la iesire vom avea singletoni.Pentru a vedea cum evolueaza sistemul vom afisa valorile obtinute in paralel cu cele apartinand sistemului de tip Mamdani.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr.  crt | Valoare densitate trafic | Valoare flow rate | Val. iesire Mamdani (extensie) | Val. iesire Sugeno (extensie) |
| 1 | 52 | 450 | 2.52 | 2.8 |
| 2 | 70 | 120 | 20 | 20 |
| 3 | 24 | 520 | -13.05 | -13.33 |
| 4 | 60 | 600 | 0 | 0 |
| 5 | 32 | 425 | -6.04 | -5.78 |
| 6 | 52 | 600 | -4.19 | -4 |
| 7 | 34 | 250 | 3.15 | 3.6 |
| 8 | 60 | 300 | 15 | 15 |
| 9 | 20 | 600 | -20 | -20 |
| 10 | 56 | 360 | 10 | 10 |

In urma modificarii tipului de system Fuzzy se observa ca valorile de mijloc variaza putin, dar de asemenea se poate observa si faptul ca limitele acestor valori (-20 si 20) nu se modifica.

1. Un al doilea studiu de caz ales este acela care implica modificarea tabelei de reguli. Noua tabela de reguli a fost aleasa in ideea de a modifica valorile de iesire cat mail in posibil. Noua tabela de reguli arata astfel:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Flow rate\ Densitate | foarte mica (VSfl) | Mica (Sfl) | Medie (Mfl) | Mare (Lfl) | foarte mare (VLfl) |
| foarte mica (VSd) | Zero | P | P | PM | PM |
| Mica (Sd) | N | Zero | P | P | PM |
| Medie (Md) | N | N | Zero | P | P |
| Mare (Ld) | NM | N | N | Zero | P |
| foarte mare (VLd) | NM | NM | N | N | Zero |

In urma modificarilor facute in cod si a simularilor realizate in tabelul urmator vor fi prezentate, tot comparativ, rezultatele obtinute in cele 2 cazuri:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr.  crt | Valoare densitate trafic | Valoare flow rate | Val. iesire Mamdani (extensie) | Val. iesire Mamdani II (extensie) |
| 1 | 52 | 450 | 2.52 | 2.52 |
| 2 | 70 | 120 | 20 | 15 |
| 3 | 24 | 520 | -13.05 | -7.3 |
| 4 | 60 | 600 | 0 | 0 |
| 5 | 32 | 425 | -6.04 | -4.2 |
| 6 | 52 | 600 | -4.19 | -4.19 |
| 7 | 34 | 250 | 3.15 | 3.15 |
| 8 | 60 | 300 | 15 | 10 |
| 9 | 20 | 600 | -20 | -10 |
| 10 | 56 | 360 | 10 | 7.58 |

Se observa ca valorile de la iesire se modifica destul de mult, modificarile fiind cu atat mai mari cu cat valorile de intrare tind spre exterior.

1. Un al treilea studio de caz imaginat consta in schimbarea functiilor de apartenenta din triunghi in trapez si observarea efectelor acestei transformari. Prin aceasta transformare cresc sansele de a avea, la un moment dat, active doar doua, sau chiar o singura regula, din tabela de reguli. De mentionat este faptul ca am modificat doar functiile de apartenenta de intrare, nu si cea de iesire. Pentru o observare mai usoara rezultatele vor fi prezentate tot comparativ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr.  crt | Valoare densitate trafic | Valoare flow rate | Val. iesire Mamdani (extensie) | Val. iesire Mamdani trapez (extensie) |
| 1 | 52 | 450 | 2.52 | 6.65 |
| 2 | 70 | 120 | 20 | 20 |
| 3 | 24 | 520 | -13.05 | -16.65 |
| 4 | 60 | 600 | 0 | 0 |
| 5 | 32 | 425 | -6.04 | -3.34 |
| 6 | 52 | 600 | -4.19 | -3.34 |
| 7 | 34 | 250 | 3.15 | 8.67 |
| 8 | 60 | 300 | 15 | 15 |
| 9 | 20 | 600 | -20 | -20 |
| 10 | 56 | 360 | 10 | 10 |

Se observa ca apar modificari destul de majore, dar valorile de minim si maxim se pastreaza.

**Concluzii**

Sistemul prezentat ofera o metoda de depasire a problemelor intampinate in trafic in marile orase ce consta in dezvoltarea si realizarea unui sistem inteligent control al semafoarelor utilizand logica Fuzzy. Astfel in functie de densitatea traficului (lungimea cozii de masini) si viteza de deplasare, durata cat semaforul va avea culoarea verde aprinsa va putea fi extinsa sau micsorata.

Pentru ca sistemul sa consume mai putine resurse necesare efectuarii calculelor este prezentat (anexa) un algoritm de eliminare a punctelor coliniare.

**Bibliografie**

* <http://en.wikipedia.org/wiki/Fuzzy_logic>
* <http://umrefjournal.um.edu.my/filebank/published_article/1689/13.pdf>
* <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.4950&rep=rep1&type=pdf>
* <http://www.google.ro/patents?hl=en&lr=&vid=USPAT6317058&id=xWgIAAAAEBAJ&oi=fnd&dq=Intelligent++traffic++control++and++warning++system++and++method&printsec=abstract#v=onepage&q=Intelligent%20%20traffic%20%20control%20%20and%20%20warning%20%20system%20%20and%20%20method&f=false>

**Anexe:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

#define min(a,b) ((a<b)?a:b)

#define max(a,b) ((a<b)?b:a)

#define nmfx1 5

#define nmfx2 5

#define nmfy 5

#define npcteR 600

float x1[nmfx1][3] = {{0, 20, 80}, {0, 20, 40}, {20, 40, 60}, {40, 60, 80}, {0, 60, 80}};

float miux1[nmfx1][3]= {{1, 0,0}, {0, 1, 0}, {0, 1, 0}, {0, 1, 0}, {0, 0, 1}};

int npx1 = 3;

float x2[nmfx2][3] = {{0, 200, 800}, {0, 200, 400}, {200, 400, 600}, {400, 600, 800}, {0, 600, 800}};

float miux2[nmfx2][3]= {{1, 0, 0}, {0, 1, 0}, {0, 1, 0}, {0, 1, 0}, {0, 0, 1}};

int npx2 = 3;

float y[nmfy][3] = {{-30, -20, -10},{-20, -10, 0},{-10, 0, 10},{0, 10, 20},{10, 20, 30}};

float miuy[nmfy][3]= {{ 0, 1, 0},{ 0, 1, 0},{ 0, 1, 0},{0, 1, 0},{ 0, 1, 0}};

int npy =3;

int reg [nmfx1][nmfx2]= {{2,1,0,0,0},{3,2,1,0,0},{4,3,2,1,0},{4,4,3,2,1},{4,4,4,3,2}};

float grad\_apartenenta (float, float\*,float\*, int);

float cog(float \*, float \*, int );

int elim\_pcte(float \*, float\*, int, float\*, float\*);

void main(void)

{

float x1c, x2c, gax1[nmfx1], gax2[nmfx2], gay[nmfy], m;

float R[npcteR+1], miuR[npcteR+1], nR[20], nmiuR[20], yinf=-30, ysup=30, step, yc; int i, j, np;

printf ("densitatea este: "); scanf ("%f",&x1c);

printf ("flow rate-ul este: "); scanf ("%f",&x2c);

for (i=0; i<nmfx1; i++)

{

gax1[i]=grad\_apartenenta(x1c, x1[i], miux1[i], npx1);

}

for (i=0; i<nmfx2; i++)

{

gax2[i]=grad\_apartenenta(x2c, x2[i], miux2[i], npx2);

}

for (i=0; i<nmfy; i++)

gay[i] =0;

//Reguli Fuzzy

for (i=0; i<nmfx1; i++)

for (j=0; j<nmfx2; j++)

if(gax1[i]!=0 && gax2[j] != 0)

{

m = min (gax1[i],gax2[j]);

if(m > gay[reg[i][j]])

gay[reg[i][j]] = m;

}

step = (ysup - yinf)/npcteR;

for (i=0, yc=yinf; i<=npcteR ; i++, yc+=step)

{

m=0;

for(j=0; j<nmfy; j++)

{

m=max(m, min(gay[j], grad\_apartenenta(yc, y[j],miuy[j], npy)));

}

miuR[i]=m;

R[i]=yc;

}

yc= cog(R, miuR, npcteR+1);

printf("Valoare iesire=%.3f\n",yc);

//defuzificare

// eliminare puncte coliniare

np = elim\_pcte(R, miuR, npcteR+1, nR, nmiuR);

yc= cog(nR, nmiuR, npcteR+1);

printf("Valoare iesire set restrans =%.3f \n",yc);

getch();

}

float grad\_apartenenta(float xc, float x[], float miux[], int npx)

{

if (xc<x[0]) return miux[0];

for (int i=0; i<npx-1;i++)

if (xc>=x[i]&&xc<=x[i+1])

{

// return (xc-min(x[i]+(miux[i]>miux[i+1]), x[i+1]))/(x[i+1]-x[i])\* (miux[i+1]-miux[i])+min (miux[i], miux[i+1]);

return (xc - x[i+(miux[i]>miux[i+1]) ]) / (x[i+1] - x[i])\* ( miux[i+1] - miux[i])+ min(miux[i],miux[i+1]);

/\*if (miux[i+1]>miux[i])

return (xc-x[i])/(x[i+1]-x[i])\*(miux[i+1]-miux[i])+miux[i];

if (miux[i+1]==miux[i])

return miux[i];

if (miux[i+1]<miux[i])

return (x[i+1]-xc)/(x[i+1]-x[i])\*(miux[i]-miux[i+1]) + miux[i+1];

\*/

}

if (xc>x[npx-1])

return miux[npx-1];

}

float cog(float y[], float miuy[], int npy)

{

float s1, s2, arie, cg, arie2, cg2;

s1=s2=0;

for(int i=0; i<npy-1; i++)

{

if(miuy[i]==miuy[i+1]) //dreptunghi

{

arie=miuy[i]\*(y[i+1]-y[i]);

cg= (y[i+1]+y[i])/2;

}

else if (miuy[i]==0 || miuy[i+1]==0) //triunghi

// if (!miuy[i]\*miuy[i+1])

{

// arie=(y[i+1]-y[i])\* (float)fabs((double)(miuy[i]-miuy[i+1]))/2;

arie=(y[i+1]-y[i])\*(miuy[i]+miuy[i+1])/2;

cg=(y[i]+y[i+1]+ (miuy[i]>miuy[i+1]?y[i]:y[i+1]))/3;

// cg=y[i]\*(1+miuy[i]>miuy) +y[i+1]\*(1+miuy[i]<miuy)/3;

}

else //trapez

{

arie=min(miuy[i],miuy[i+1])\*(y[i+1]-y[i]);

cg= (y[i+1]+y[i])/2;

// arie2=(y[i+1]-y[i])\*(float)fabs((double)(miuy[i]-miuy[i+1]))/2;

arie2=(y[i+1]-y[i]) \* (max(miuy[i], miuy[i+1])-min(miuy[i], miuy[i+1]))/2;

//cg=(y[i]+y[i+1]+ (miuy[i]>miuy[i+1]?y[i]:y[i+1]))/3;

cg2=(y[i]+y[i+1]+ (miuy[i]>miuy[i+1]?y[i]:y[i+1]))/3;

cg=(arie\*cg+arie2\*cg2)/(arie+arie2);

arie+=arie2;

}

// printf("Arie%d= %f, cg%d= %f\n", i+1, arie, i+1, cg);

s1+=arie\*cg;

s2+=arie;

}

return s1/s2;

}

int elim\_pcte(float R[], float miuR[], int p, float nR[], float nmiuR[])

{

int np, i;

float x1, x2, x3, y1, y2, y3, m1, m2;

x1 = R[0];

y1 = miuR[0];

x2 = R[1];

y2 = miuR[1];

x3 = R[2];

y3 = miuR[2];

np=0;

for(i=3 ; i<p; i++)

{

if(fabs((y2-y1)/(x2-x1)-(y3-y2)/(x3-x2)) > 0.00001)

{

nR[np] = x1;

nmiuR[np] = y1;

np++;

x1 = x2; y1 = y2;

}

x2 = x3;

y2 = y3;

x3 = R[i];

y3 = miuR[i];

}

nR[np] = x1;

nmiuR[np] = y1;

np++;

if(fabs((y2-y1)/(x2-x1)-(y3-y2)/(x3-x2)) > 0.00001)

{

nR[np] = x2;

nmiuR[np] = y2;

np++;

}

nR[np] = x3;

nmiuR[np] = y3;

//np++;

// printf("\n");

for(i=0;i<=np;i++)

printf("(%.1f, %.2f)\n",nR[i] , nmiuR[i]);

return np;

}