UNIVERSITATEA „SAPIENTIA” DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,

TÎRGU MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

CONTROLUL TRAFICULUI URBAN IMPLEMENTAT CU SISTEM DE REGLARE FUZZY

PROIECT DE DIPLOMĂ

Coordonator științific:

Prof.dr.ing. Dávid László,

Prof.dr. Farkas Csaba

Absolvent:

Miklo Jozsef-Péter

2024

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA **Viza facultății:**
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Specializarea: **Automatică și informatică aplicată**

LUCRARE DE DIPLOMĂ

Coordonator științific:	Candidat: **Miklo Jozsef-Péter**
Prof.dr.ing. Dávid László, Prof.dr. Farkas Csaba	Anul absolvirii: **2024**

a) Tema lucrării de licență:
CONTROLUL TRAFICULUI URBAN IMPLEMENTAT CU SISTEM DE REGLARE FUZZY

b) Problemele principale tratate:
- Studiu privind sistemele de reglare aplicate în sisteme de control a traficului urban
- Studiu teoretic al stabilității sistemelor de trafic urban pe baza unui model neliniar
- Realizarea unei aplicații pentru simularea procesului studiat
- Clasificarea metodelor și studiul rezultatelor obținute pe baza metodelor de reglare studiate

c) Desene obligatorii:
- Schema bloc al aplicației
- Diagrame UML privind software-ul realizat.

d) Softuri obligatorii:
-Aplicație SUMO, reglare fuzzy intersecții individuale, reglare fuzzy pentru corelarea diferitelor intersecții adiacente

e) Bibliografia recomandată:

- M. Bando and K. Hasebe, A. Nakayama, A. Shibata, Y. Sugiyarna, “Dynamical model of traffic congestion and numerical simulation”, 1995
- Rui Jiang, Qingsong Wu, Zuojin Zhu, “Full velocity difference model for a car-following theory”, 2001
- Shaowei Yu , Qingling Liu, Xiuhai Li, “Full velocity difference and acceleration model for a car-following theory”, 2012
- Javed Alam, Dr. M. K. Pandey, Husain Ahmed, “Development of Traffic Light Control System for Emergency Vehicle Using Fuzzy Logic”, 2013
- Dávid L, Márton L., Rețele Neuronale Artificiale şi Logica Fuzzy în Automatizări, Editura Universităţii “Petru Maior”, 2000, ISBN 973-8084-02-4
- Márton Lőrinc, Irányítsátechnika, Scientia, 2009

f) Termene obligatorii de consultații: săptămânal
g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca,
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș
Primit tema la data de: 15.05.2023
Termen de predare: 10.07.2024

Semnătura Director Departament — Semnătura coordonatorului

Semnătura responsabilului programului de studiu — Semnătura candidatului

Declarație

Subsemnata/ul ....., absolvent(ă) al/a specializării ……., promoția…… cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea,

Data: Absolvent

Semnătura……

Ide kerül a Turnitin similarity report

Controlul traficului urban implementat cu sistem de reglare fuzzy

Extras

În prezent, un număr foarte mare de vehicule circulă în orașe și pe autostrăzi. Atunci când numărul de vehicule depășește capacitatea drumurilor, se formează blocaje de trafic. Pentru a atenua aceste blocaje, au apărut sisteme moderne de gestionare a traficului. Scopul lucrării este evitarea formării ambuteiajelor, reducerea timpului de călătorie sau atenuarea blocajelor deja formate. De-a lungul anilor, au apărut mai multe abordări pentru gestionarea traficului, printre care: inteligența artificială, controlul fuzzy și controlul predictiv al modelelor.

În această lucrare, am studiat diferite modele matematice dinamice a căror stabilitate determină formarea blocajelor. În plus, am realizat simularea unor modele matematice în mediul Matlab. Simularea traficului urban și gestionarea traficului au fost realizate cu ajutorul software-ului SUMO (Simulation of Urban Mobility). Acesta este un software de simulare open-source, utilizat pentru modelarea și analiza rețelelor de transport. TraCI (Traffic Control Interface) este un protocol care permite aplicațiilor externe să comunice cu SUMO. Cu ajutorul TraCI, am putut stabili o conexiune între SUMO și fișierele Python, astfel încât am creat un control fuzzy propriu într-un fișier Python, cu care ulterior am controlat semafoarele din intersecțiile din simulare în mod corespunzător traficului. Ulterior, am construit încă un controlor fuzzy care a jucat un rol în sincronizarea intersecțiilor adiacente, creând un efect de undă verde pe un traseu selectat. La sfârșitul lucrării am efectuat măsurători experimentale în intersecțiile controlate de semafoare pentru a demonstra eficiența traficului controlat cu ajutorul controlului fuzzy.

Cuvinte cheie: gestionarea traficului, Fuzzy, SUMO, Python, undă verde

SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK

FUZZY SZABÁLYOZÁSSAL MEGVALÓSÍTOTT VÁROSI FORGALOMIRÁNYÍTÓ RENDSZER

DIPLOMADOLGOZAT

Témavezető:
Dr. Dávid László, egyetemi tanár
Dr. Farkas Csaba, egyetemi tanár

Végzős hallgató:
Miklo Jozsef-Péter

2024

Kivonat

Napjainkban nagyon sok jármű közlekedik városokban és autópályákon. Amikor a járművek száma meghaladja az utak kapacitását forgalmi torlódások alakulnak ki. Ezen torlódások enyhítésére modern forgalomirányítási rendszerek jelentek meg. A dolgozat célja a forgalmi dugók kialakulásának elkerülése, az utazási idő csökkentése vagy a már kialakult torlódások enyhítése. Az évek során többféle megközelítés született a forgalomirányításra, többek között: mesterséges intelligenciával, fuzzy szabályozással, modell prediktív irányítással.

A dolgozatban különböző dinamikus matematikai modelleket tanulmányoztunk, amelyek stabilitása meghatározza a torlódások kialakulását. Emellett elvégeztük néhány matematikai modell szimulációját is Matlab környezetben. A városi forgalom szimulációját és a forgalomirányítást egy SUMO (Simulation of Urban MObility) szoftverrel végeztük. Ez egy nyílt forráskódú szimulációs szoftver, amelyet közlekedési hálózatok modellezésére és elemzésére használnak. A TraCI (Traffic Control Interface) egy olyan protokollt biztosít, amely lehetővé teszi, hogy a külső alkalmazások kommunikáljanak a SUMO-val. A TraCI segítségével egy kapcsolatot tudunk létrehozni a SUMO és a Python fájlok között, így egy Python fájlban létrehoztuk a saját fuzzy irányításunkat, amivel később a szimulációban lévő kereszteződések jelzőlámpáit irányítottuk a forgalomnak megfelelő módon. Ezt követően még egy fuzzy szabályozót építettünk fel, ami a szomszédos kereszteződések összehangolásában játszott szerepet, kialakítva egy zöldhullám effektust egy kiválasztott útvonalon. A dolgozat végén kísérleti méréseket végeztünk a jelzőlámpákkal irányított kereszteződésekben, hogy bebizonyítsuk a fuzzy szabályozással irányított forgalom hatékonyságát.

Kulcsszavak: forgalomirányítás, Fuzzy, SUMO, Python, zöldhullám

Abstract

Nowadays, a large number of vehicles travel in cities and on highways. When the number of vehicles exceeds the capacity of the roads, traffic congestion occurs. To mitigate these congestions, modern traffic management systems have emerged. The aim of this thesis is to avoid the formation of traffic jams, reduce travel time or alleviate already formed congestions. Over the years, several approaches have been developed for traffic management, including: artificial intelligence, fuzzy control, and model predictive control.

In this thesis, we studied various dynamic mathematical models whose stability determines the formation of congestions. Additionally, we performed simulations of some mathematical models in a Matlab environment. The urban traffic simulation and traffic management were carried out using SUMO (Simulation of Urban Mobility) software. This is an open-source simulation software used for modeling and analyzing transportation networks. TraCI (Traffic Control Interface) is a protocol that allows external applications to communicate with SUMO. Using TraCI, we were able to establish a connection between SUMO and Python files, thus creating our own fuzzy control in a Python file, with which we later controlled the traffic lights at intersections in the simulation appropriately to the traffic. Subsequently, we built another fuzzy controller that played a role in synchronizing adjacent intersections, creating a green wave effect on a selected route. At the end of the thesis, we conducted experimental measurements at the traffic light-controlled intersections to demonstrate the efficiency of traffic controlled by fuzzy regulation.

Keywords: traffic management, Fuzzy, SUMO, Python, green wave

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. Bevezető

Régen az utcák és az utak passzív infrastruktúrák voltak. Jelentős céljuk az volt, hogy gyors és kényelmes vezetést biztosítsanak. Azonban manapság az utak már sok helyen nem felelnek meg ennek a célnak. Az utóbbi évtizedekben jelentősen megnőtt az járművek száma, ami forgalmi dugókhoz és balesetekhez vezetett. Így kezdetben fix időzítésű vezérlőlámpákat, majd később számítógépes programok által vezérelt lámpákat vezettek be. [1]

A forgalmi dugókkal kapcsolatos problémák enyhítésére különböző megoldásokkal is próbálkoznak: új utak építése, út díjak kivetése, tömegközlekedés előmozdítása, vagy a meglévő infrastruktúra hatékonyabb kihasználása.

Az intelligens városi közlekedés területén számos elméleti és technológiai innováció és alkalmazás tanúja voltunk, köztük a forgalmi irányítás, amit az intelligens közlekedés koronájának tartanak. Ez kulcsfontosságú intézkedésként szolgált a forgalmi dugók enyhítésére és a forgalmi problémák megoldására. Ennek eredményeként olyan fejlett forgalmi jel irányítási rendszerek jelentek meg, mint például a SCOOT[1], SCATS[2] és a modellezés-alapú algoritmusok, adatvezérelt algoritmusok és mesterséges intelligencia alapú kiváló forgalmi irányítási algoritmusok, amelyek szinte egy évszázad fejlesztésének során jöttek létre, és támogatták az urbanizáció gyors fejlődését. [2]

A városi hálózatok forgalomirányító felépítésük szerint a következő kategóriákba sorolhatók: centrális, elosztott (decentralizált) és vegyes. A centrális forgalomirányító esetében minden döntést egy központi gép hoz, amit később továbbít a terepi berendezéseknek. Az elosztott architektúra eseténél központi gép nélkül van megvalósítva az irányítás, a terepi gépek elosztják egymás között a számításokat. Az elosztott és a vegyes irányítási architektúrák kevésbé elterjedtek, a centrális architektúrához viszonyítva. Viszont az utóbbi két architektúra előnye, hogy nem áll fent a központi gépről való leszakadás veszélye és nagyobb biztonsági üzemelés valósítható meg velük. Alkalmazási példaként az elosztott irányításási rendszerekre megemlíthető az ausztráliai SCATS és az Európában működő Utopia[3]. [3]

A dolgozat két fő részből tevődik össze:

1. forgalmi áramlás matematikai modellezése, stabilitásvizsgálata és a modell szimulációja

[1] https://trlsoftware.com/products/traffic-control/scoot/

[2] https://www.scats.nsw.gov.au/home

[3] https://www.trafitek.com/atc-utopia

2. kereszteződésekben lévő jelzőlámpák Fuzzy irányítása szimulációs szoftverben

A dolgozat első részében különböző dinamikus modelleket hasonlítunk össze, stabilitásvizsgálatot végzünk és ezenkívül egy modellnek a szimulációját is elvégezzük. A második részben egy szimulációs program segítségével Marosvásárhely területén Fuzzy irányítással irányítjuk a forgalmat. Ebben a részben kapunk egy betekintést a szimulációs szoftverbe, a szimuláció indításáról Python programozási nyelvben. Ezek után röviden ismertetjük a Fuzzy logikát és Fuzzy szabályozást és bemutatjuk az általunk felépített Fuzzy irányítást. Majd egy gyakorlati megvalósítását is láthatjuk a Fuzzy irányításnak, ahol a jelzőlámpák zöld idejének időtartamát irányítottuk. A dolgozat végén pedig méréseket végeztünk, hogy meggyőződjünk a forgalomirányítás hatékonyságáról és következtetéseket vontunk le a tanulmányozott témáról.

2. Elméleti megalapozás és szakirodalmi tanulmány

2.1. Közlekedési áramlás matematikai modellezése

A közlekedési dinamika egyik legjelentősebb problémája a forgalmi torlódások kialakulása. A torlódásokat balesetek, közlekedési lámpák vagy az utak túlterhelése okozhatja, amik instabilitást hoznak a rendszerbe. Ezek tanulmányozására egy dinamikus rendszermodellt vezetünk be, ami a közlekedési áramlást dinamikusan modellezi.

Vegyünk egy egyszerű modellt, amit M. Bando és társai (1995) cikkében találunk OVM (Optimal Velocity Model) néven [4]. Ebben a modellben nem vesszük figyelembe a járművek hosszúságát és a vezetők jellemét sem. Tehát minden autóvezető ugyanolyan érzékenységgel vezet. Feltételezzük, hogy minden jármű a legális V sebességgel halad és reagál az előtte lévő jármű távolságára. Így fékezéssel vagy gázolással szabályozhatjuk a jármű gyorsulását a következő képlet alapján:

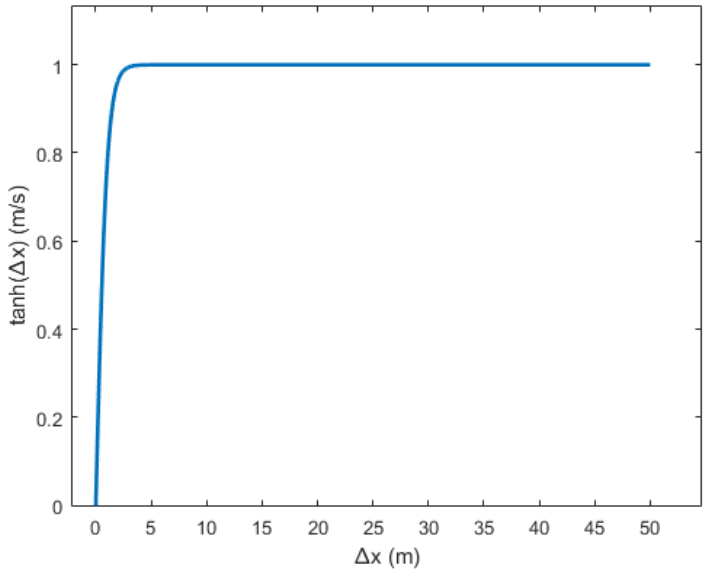
$$\ddot{x_n}(t) = \alpha[V(\Delta x_n(t)) - \dot{x_n}(t)] \tag{1}$$

ahol $\ddot{x_n}(t)$ jelöli az n-dik autó gyorsulását, α-val jelöltük a vezető érzékenységét $\alpha \in [0,1]$, $\Delta x_n(t) = x_{n+1}(t) - x_n(t)$, jelöli az (n+1)-dik és az n-dik jármű közötti távolságot, $V(\cdot)$ jelöli a sebességfüggvényt és $\dot{x_n}(t)$ pedig az n-dik jármű sebességét.

A sebességfüggvény a következő egyszerű képlettel adhatjuk meg:

$$V(\Delta x) = \tanh(\Delta x) \tag{2}$$

ahol Δx a járművek közötti távolságot jelöli.



2.1.1. ábra – Sebességfüggvény ábrázolása

A 2.1.1-es ábrán láthatjuk a sebességfüggvény ábrázolását, ahogyan a Δx változik 0-tól 50-ig. Megfigyelhetjük, hogy a függvény monoton növekvő és 1-ben van a felső korlátja.

Egy következő modellt is vegyünk szemügyre, amit Rui Jiang és munkatársai (2001) foglaltak össze és nevezték FVDM(Full Velocity Difference Model) modellnek [5]. Ebben a modellben figyelembe veszik mind a pozitív, mind a negatív sebességkülönbségeket és határozzák meg az autó gyorsulását. Ezt a modellt a következő alakban írhatjuk le:

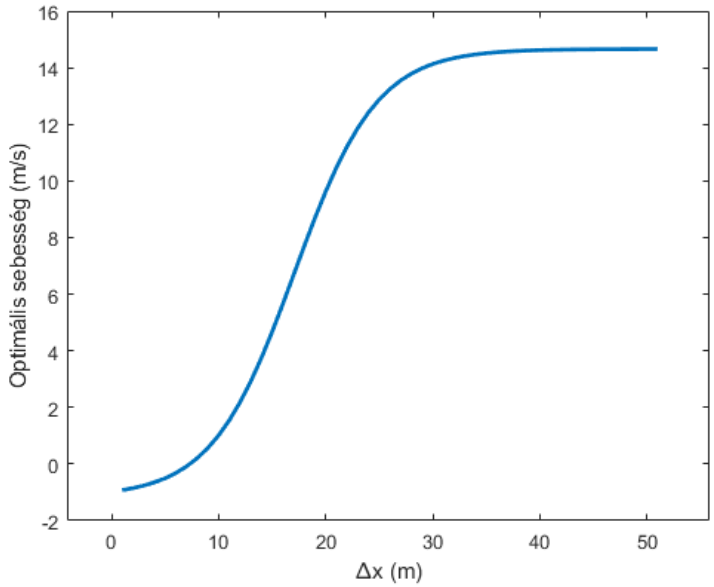
$$\ddot{x_n}(t) = \alpha\left[V\left(\Delta x_n(t)\right) - v_n(t)\right] + \lambda \Delta v_n(t) \tag{3}$$

ahol paraméterek megfelelnek az (1) képletben lévő paramétereknek, és még kiegészülnek a következőkkel: $v_n(t)$ jelöli az n-dik jármű sebességét, $\Delta v_n(t) = v_{n+1}(t) - v_n(t)$ jelöli az (n+1)-dik és az n-dik jármű sebességkülönbségét, λ a vezető érzékenységi együtthatója a sebességkülönbséghez $\lambda \in [0,1]$, V(.) az optimális sebességfüggvény.

Az optimális sebességfüggvényt a Dirk Helbing és Benno Tilch által javasolt alakban írhatjuk fel [7]:

$$V(\Delta x_n) = V_1 + V_2 \tanh(C_1(\Delta x_n - l_c) - C_2) \tag{4}$$

ahol Δx_n jelöli a már fentebb említett két autó közötti távolságot, l_c jelöli a vezető autó hosszát, V_1 egy eltolást, míg V_2 egy skálázást jelöl, C_1 a távolság együtthatója és C_2-vel állítjuk be a megfelelő paramétert a tangh() függvénynek.



2.1.2. ábra – Optimális sebességfüggvény ábrázolása

Tekintsük meg a 2.1.2. ábrát, ami szemlélteti az optimális sebességfüggvényt Δx_n függvényében. A használt paraméterek Stuttgart forgalmára voltak jellemzőek, amiket a

következőképpen választottak meg: $V_1 = 6.75\ \frac{m}{s}, V_2 = 7.91\ \frac{m}{s}, l_c = 5m, C_1 = 0.13\ m^{-1}, C_2 = 1.57$.

Megfigyelhetjük, hogy ezen az ábrán már egy teljes tangens hiperbolikus függvény látható eltolva és skálázva, ami már valóságosabban ábrázolja a sebességet, mint a Bando és társai cikkében [4] leírt sebesség, ahol csak a két jármű közötti távolságot vették figyelembe.

A FVDM modellnek egy továbbfejlesztett változatát készítették el Shaowei Yu és társai (2012), amit FVDAM (Full Velocity Difference and Acceleration Model) modellnek neveztek [6]. A FVDM és a FVDAM közötti lényeges különbség az, hogy a FVDAM modell figyelembe veszi a vezető (előtte levő) autó gyorsulását,tehát ha az aktuális autó sebessége nagyobb az előtte lévőnél, viszont az előtte lévő gyorsabb, akkor nem fog rögtön fékezni a követő autó, még ha biztonságos követési távolságot meg is haladja.

$$\ddot{x_n}(t) = \alpha\left[V\left(\Delta x_n(t)\right) - v_n(t)\right] + \lambda\Delta v_n(t) + k a_{n+1}(t) \tag{5}$$

ahol a jelölések megegyeznek az (1) és (2) egyenletekben használt paraméterekkel, kiegészítve $a_{n+1}(t)$-vel, ami jelöli az (n+1)-dik jármű (vezető) gyorsulását, k pedig a követő autó érzékenységi együtthatóját jelöli, $k \in [0,1]$. Ha k=0 akkor a modell FVDM modell, ha k>0 akkor FVDAM modellel dolgozunk.

Ugyancsak a FVDM modellből kiindulva Jing Zhang és munkatársai (2019) létrehoztak egy modellt, ami előrejelzi az vezető jármű viselkedését [8]. A modellt az (6) egyenlet mutatja be:

$$\ddot{x_n}(t) = \alpha\{V[\Delta x_n(t) - \beta(\Delta x_n(t+\tau) - \Delta x_n(t))] - \Delta v_n(t)\} + \lambda\Delta v_n(t) \tag{6}$$

Az (6) egyenletet Taylor sorbafejtéssel leegyszerűsíthetjük a következőképpen: $\Delta x_n(t) = \Delta x_n(t) + \tau\Delta v_n(t)$, amiből kapjuk a (7) egyenletet:

$$\ddot{x_n}(t) = \alpha\left[V\left(\Delta x_n(t) + \beta\tau\Delta v_n(t)\right) - v_n(t)\right] + \lambda\Delta v_n(t) \tag{7}$$

ahol a távolságok, sebességek és sebességkülönbségek jelölése megegyezik a fent említett modellek jelölésével. Bővítésként megjelenik a β ami egy erősségi együttható és τ egy prediktív időtartam.

A bemutatott négy modell közül a FVDAM modellt választottuk a közlekedési áramlás modellezésére, mert ez a modell nemcsak a sebességkülönbséget, hanem még az vezető jármű gyorsaságát is figyelembe veszi a modellezésben.

2.2. A dinamikus modellek stabilitásvizsgálata

A dinamikus modell meghatározása után kulcsfontosságú a modell stabilitásának vizsgálata. A stabilitás az egyik alapvető tényező a modell megbízhatósága és hatékonysága szempontjából. Ha a modell instabil, akkor a rendszerben váratlan és kiszámíthatatlan jelenségek is történhetnek, amik veszélyeztethetik a közlekedést és növelhetik a balesetek kockázatát.

A nemlineáris rendszerek stabilitás vizsgálatára a Popov kritérium mellett a direkt és az indirekt Lyapunov módszerek a legelterjedtebbek. A direkt Lyapunov módszer a dinamikus rendszerhez rendel egy Lyapunov függvénynek nevezett energiafüggvényt, majd az energiafüggvény változásából von le következtetéseket a rendszer stabilitásáról. Legyen $\dot{\underline{x}} = f(\underline{x}, t)$ egy nemlineáris dinamikus modell és $f(\underline{0}, t)$=0 minden $t \geq 0$ –ra. Ha létezik egy olyan $V(\underline{x}, t)$ Lyapunov függvény, amely deriválható $\underline{x} = \underline{0}$ egyensúlyi pont körül és az alábbi feltételeket teljesíti, akkor beszélhetünk stabil rendszerről:

1. $V(\underline{x}, t)$ pozitív definit: $V(\underline{0}, t) = 0$ és $V(\underline{x}, t) > 0, ha\ x \neq 0$
2. $\dot{V}(\underline{x}, t)$ negatív szemidefinit: $\dot{V}(\underline{x}, t) \leq 0$
3. $\dot{V}(\underline{x}, t)$ negatív definit: $\dot{V}(\underline{x}, t) < 0$

Ha az 1. és a 2. feltétel teljesül, akkor $\underline{x} = \underline{0}$ egyensúlyi pontja a rendszernek stabil és ha az 1. és 3. feltétel teljesül akkor $\underline{x} = \underline{0}$ egyensúlyi pont aszimptotikusan stabil.

Az indirekt Lyapunov módszer esetén a nemlineáris rendszert linearizáljuk, hogy meghatározzuk a nemlineáris rendszer helyi stabilitását. Vegyük az előbbi példaként vett nemlineáris dinamikus modellt: $\dot{\underline{x}} = f(\underline{x}, t)$, $f(\underline{0}, t)$=0 minden $t \geq 0$ –ra. Legyen A(t) $= \frac{\partial f}{\partial x}\Big|_{x=0}$ az $f(\underline{x}, t)$ Jacobi mátrixa, melyet x=0 kezdőpontban értékelünk ki. Minden t-re kapunk egy maradékot, amit a következőképpen írhatunk le: $f_1(\underline{x}, t) = f(\underline{x}, t) - \mathrm{A(t)}\underline{x}$. Mivel a maradék nem biztos, hogy egyenletesen közelít a 0-hoz, ezért egy erősebb feltételre van szükség: $\lim_{\|\underline{x}\| \to 0} sup_{t \geq 0} \frac{\|f_1(\underline{x},t)\|}{x\|\underline{x}\|} = 0$. Ha ez az egyenlet teljesül, akkor $\dot{z} = A(t)z$ rendszer az egyenletes linearizálása az eredeti rendszernek az origó körül. Amikor a linearizáció létezik akkor annak a stabilitása meghatározza az eredeti nemlineáris rendszer helyi stabilitását.

Azért hogy összehasonlítsuk a szakirodalomban lévő modellek stabilitásvizsgálatával mi az indirekt Lyapunov elvet fogjuk alkalmazni a stabilitásvizsgálatra. Ha a linearizált modell stabil, akkor a nemlineáris is, csak nem tudjuk meghatározni, hogy milyen környezetben fog stabil maradni.

Először is vizsgáljuk meg az első, (1) egyenletben meghatározott modell lineáris stabilitását. Legyen az egyenletes állandósult állapotbeli áramlás a következő:

$$\mathrm{x_n^0(t) = bn + V(b)t,\ \ b = \frac{L}{N}} \tag{8}$$

ahol b jelöli a két jármű közötti állandó távolságot, n=0,1,...N, ahol N jelöli az összes járművek számát, L jelöli az út hosszát és V(b) az optimális sebességet jelöli. Ahhoz, hogy a (8) egyenletet linearizáljuk szükséges alkalmaznunk a perturbáció módszerét, ahol $y_n(t)$ egy kis zavar amit hozzáadunk a stabil egyenlethez.

$$x_n(\mathrm{t}) = \mathrm{x_n^0(t)} + y_n(t), \qquad |y_n(t)| \ll 1 \tag{9}$$

A (8) és (9) egyenletet behelyettesítve az (1) egyenletbe, majd leegyszerűsítve $\Delta x_n(t) = \Delta \mathrm{x_n^0(t)} + \Delta y_n(t) = b(n+1) + V(b)t - bn - V(b)t = b + \Delta y_n(t)$ és $\dot{x_n}(t) = V(b) + \dot{y_n}(t)$ megkapjuk a következő összefüggést:

$$\ddot{y_n}(t) = \alpha\left[V\left(b + \Delta y_n(t)\right) - V(b) - \dot{y_n}(t)\right] \tag{10}$$

Majd ezek után Taylor sorba fejtve az (10) egyenlet leegyszerűsödik a következőképpen:

$$\ddot{y_n}(t) = \alpha[V'(b)\Delta y_n(t) - \dot{y_n}(t)] \tag{11}$$

$y_n(t)$ Fourier transzformáció után a következővel helyettesítve: $y_n(t) = e^{i\alpha_k n + zt}$, a deriváltak pedig: $\dot{y_n}(t) = ze^{i\alpha_k n + zt}$, $\ddot{y_n}(t) = z^2 e^{i\alpha_k n + zt}$, amit behelyettesítve a (11) egyenletbe majd az egész egyenletet elosztva $e^{i\alpha_k n + zt}$-el a következő összefüggést kapjuk:

$$\mathrm{z^2} + \alpha z - \alpha V'(b)(e^{i\alpha_k} - 1) = 0 \tag{12}$$

ahol $\mathrm{z = u + vi}$ egy komplex számot jelöl, $\alpha_k = \frac{2\pi}{N}k,\ k = 0, 1, 2, \dots, N-1$.

Azért szükséges a (12)-es egyenlet előtt Fourier transzformációval számoljunk Laplace transzformáció helyett, mert egy periodikus jelről beszélünk. A Laplace transzformált esetében egy kezdőpontból indulunk ki, ahol az integrál [0, +∞) intervallumon van értelmezve, míg a Fourier transzformáltat periodikus jelek vizsgálatára alkalmazzák, ahol nincs kezdőpont, tehát az integrál (–∞, +∞) intervallumon van értelmezve.

A (12)-es egyenletet megoldva megkapjuk a stabilitási feltételt:

$$V'(b) < \frac{\alpha}{2} \tag{13}$$

Ha teljesül a (13)-as feltétel, akkor a felírt dinamikus modell stabil, ha $V'(b) = \frac{\alpha}{2}$ akkor a modell a stabilitás határán van és ha $V'(b) > \frac{\alpha}{2}$ akkor a modell instabil.

Hasonlatosképpen vizsgáljuk meg a FVDM modell lineáris stabilitását és hasonlítsuk össze miben tér el az előzőtől. A (8)-(9) egyenletek nem változnak, viszont a (10) egyenlet a következőképpen alakul: $\ddot{y_n}(t) = \alpha\left[V\left(b + \Delta y_n(t)\right) - V(b) - \dot{y_n}(t)\right] + \lambda\Delta\dot{y_n}(t)$, majd Taylor sorbafejtés után a (11) egyenlet helyett a következőt kapjuk:

$$\ddot{y_n}(t) = \alpha[V'(b)\Delta y_n(t) - \dot{y_n}(t)] + \lambda\Delta\dot{y_n}(t) \tag{14}$$

Előzőhöz képest még bejött $\lambda\Delta\dot{y_n}(t)$ tag az egyenletbe. A (12) egyenlet pedig a következőképpen változik meg:

$$\mathrm{z}^2 + \alpha z - (e^{i\alpha_k} - 1)(\alpha V'(b) + \lambda \mathrm{z}) = 0 \tag{15}$$

Az előbbi modellhez képest itt még bejött egy $-\lambda z(e^{i\alpha_k} - 1)$ tag az egyenletbe. Behelyettesítve $z = vi$ és $\mathrm{e}^{\mathrm{i}\alpha_\mathrm{k}} = \cos\alpha_k + i\sin\alpha_k$ a (15) egyenletbe kapjuk a következő egyenletet:

$$-v^2 - \alpha V'(b)(\cos\alpha_k - 1) + \lambda v\sin\alpha_k + i(-\alpha V'(b)\sin\alpha_k - \lambda v(\cos\alpha_k - 1) + \alpha v) = 0 \tag{16}$$

Innen pedig megkapjuk a stabilitási feltételt:

$$V'(b) < \frac{\alpha}{2} + \lambda \tag{17}$$

Itt a stabilitáshoz már hozzájárul a λ is, ami az előző modellben még nem jelent meg.

A FVDAM modell stabilitásvizsgálata is eltér néhány helyen a FVDM-hez képest. A (8)-(9) egyenletek ebben az esetben is megmaradnak, a (10) egyenlet pedig a következőképpen alakul: $\ddot{y_n}(t) = \alpha\left[V\left(b + \Delta y_n(t)\right) - V(b) - \dot{y_n}(t)\right] + \lambda\Delta\dot{y_n}(t) + k\ddot{y_{n+1}}(t)$, Taylor sorbafejtés után pedig:

$$\ddot{y_n}(t) = \alpha[V'(b)\Delta y_n(t) - \dot{y_n}(t)] + \lambda\Delta\dot{y_n}(t) + k\ddot{y_{n+1}}(t) \tag{18}$$

Látható, hogy itt már bejött két plusz tag az első modellhez képest és egy plusz tag a FVDM modellhez képest ($k\ddot{y_{n+1}}(t)$). $y_n(t) = e^{i\alpha_k n + zt}$ behelyettesítve a modellbe a következőképpen alakul az egyenlet:

$$\mathrm{z}^2(1 - ke^{i\alpha_k}) + \alpha z - (e^{i\alpha_k} - 1)(\alpha V'(b) + \lambda \mathrm{z}) = 0 \tag{19}$$

A FVDM modellhez viszonyítva a z^2 meg lett szorozva még $-\mathrm{ke}^{\mathrm{i}\alpha_\mathrm{k}}$-val.

Behelyettesítve $z = vi$ és $\mathrm{e}^{\mathrm{i}\alpha_\mathrm{k}} = \cos\alpha_k + i\sin\alpha_k$ a (19) egyenletbe kapjuk a következő egyenletet:

$$v^2 k\cos\alpha_k - v^2 - \alpha V'(b)(\cos\alpha_k - 1) + \lambda v\sin\alpha_k + i(v^2 k\sin\alpha_k - \alpha V'(b)\sin\alpha_k - \lambda v(\cos\alpha_k - 1) + \alpha v) = 0 \tag{20}$$

Innen pedig megkapjuk a semleges stabilitási feltételt:

$$V'(b) < \frac{(2\lambda+\alpha)(4\lambda k+\alpha k+\alpha)}{2\alpha(1-k)^2} = \frac{\alpha}{2(1-k)^2} + \frac{\lambda}{(1-k)^2} + \frac{3\alpha k\lambda+4k\lambda^2}{(1-k)^2} \quad (21)$$

Itt már teljesen megváltozik a stabilitási feltétel, de k=0-ra $V'(b) < \frac{\alpha}{2} + \lambda$ a FVDAM leegyszerűsödik FVDM modellre.

Végül pedig vizsgáljuk meg a Jing Zhang és munkatársai által bemutatott modellt, amit a (6) egyenlet mutatott be. Az állandósult állapotbeli áramlás (8) és a perturbációs módszer (9) egyenletek ebben az esetben sem változnak. A (10) egyenlet itt a következőképpen alakul: $\ddot{y_n}(t) = \alpha[V(b + \Delta y_n(t) + \beta\tau\dot{y_n}(t)) - V(b) - \dot{y_n}(t)] + \lambda\Delta\dot{y_n}(t)$, majd Taylor sorba fejtés után a következőképpen alakul az egyenlet:

$$\ddot{y_n}(t) = \alpha\{V'(b)[\Delta y_n(t) + \beta\tau\Delta\dot{y_n}(t)] - \dot{y_n}(t)\} + \lambda\Delta\dot{y_n}(t) \quad (22)$$

Jing Zhang és munkatársai cikkét tanulmányozva egy hibára bukkantunk. A (22) egyenlet náluk helytelenül szerepel:$\beta\tau\Delta\dot{y_n}(t)$ helyett $\beta\tau\dot{y_n}(t)$ szerepel, amivel a későbbi eredményeket sem tudjuk megkapni.

$y_n(t) = e^{i\alpha_k n+zt}$ behelyettesítve a (22) egyenletbe következőt összefüggést kapjuk:

$$z^2 + \alpha z - (e^{i\alpha_k} - 1)(\alpha V'(b) + \lambda z + \alpha\beta\tau V'(b)z) \quad (23)$$

FVDM modellnél a (15) egyenlettel összehasonlítva ebben az egyenletben az $-(e^{i\alpha_k} - 1)$ még meg van szorozva $\alpha\beta\tau V'(b)z$-vel.

$z = vi$ és $e^{i\alpha_k} = \cos\alpha_k + i\sin\alpha_k$ behelyettesítve a (23) egyenletbe megkapjuk a stabilitási feltételt, ami ebben az esetben a következő:

$$\alpha < \frac{2(V'(b) - \lambda)}{1 + 2\beta\tau V'(b)} \rightarrow V'(b) < \frac{-2\lambda - \alpha}{2\alpha\beta\tau - 2} \quad (24)$$

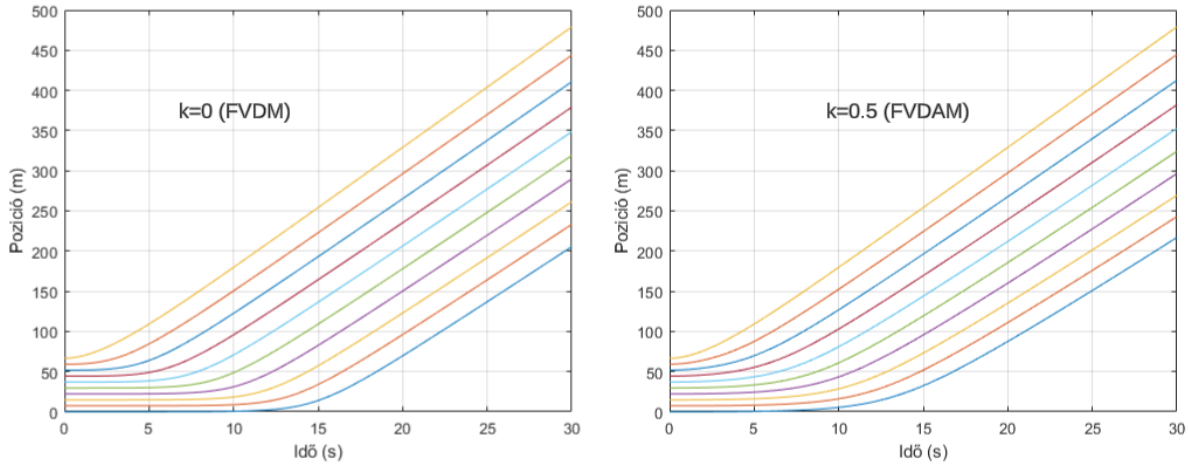
2.3. FVDAM és FVDM modellek szimulálása

Ahhoz, hogy meggyőződjünk a modellek helyes működéséről szükséges szimulációkat végeznünk a modellekről. A modellek közül a FVDAM és FVDM modellek szimulációját végeztük el Matlab környezetben. A szimuláció következőképpen zajlik: a közlekedési jelzőlámpa piros és 10 autó várakozik a piros lámpánál 7.4 m követési távolsággal egymástól. A lámpa $t = 0$ –ban zöldre vált és az autók elindulnak.

A szimuláció során vizsgáljuk az autók pozícióját, sebességét és gyorsulását idő függvényében. A szimulációt Shaowei Yu és társai [6] által használt paraméterekkel végeztük, amiket következőképpen választottak meg: $l_c = 5\,m,\ \alpha = 0.41\,s^{-1},\ \lambda = 0.5,\ V_1 = 6.75\,m/s,$

$V_2 = 7.91\, m/s$, $C_1 = 0.13\, m^{-1}$, $C_2 = 1.57$. Létrehoztunk egy "V()" optimális sebesség függvényt, ami paraméterként kapott két autó közötti távolságból meghatározza az optimális sebességet. A legelső (vezető) jármű gyorsulását egy exponenciálisan csökkenő függvénnyel adtuk meg. Ezek után létrehoztunk egy függvényt, amiben az autók differenciálegyenleteit felírtuk a paraméterek és az előbbi két függvény segítségével. A "Matlab"-ba beépített "ode45()" általános differenciálegyenletek megoldására szolgáló függvény segítségével oldjuk meg az előbbi függvényben leírt autók differenciálegyenleteit. Az "ode45()" függvénynek paraméterként megadjuk a függvényt, amiben a differenciálegyenletek vannak leírva, az időtartományt és minden autó kezdeti pozícióját és sebességét. Minden autó kezdeti sebessége 0, mivel piros lámpánál várakoznak és 7.4 m távolságra vannak egymástól.

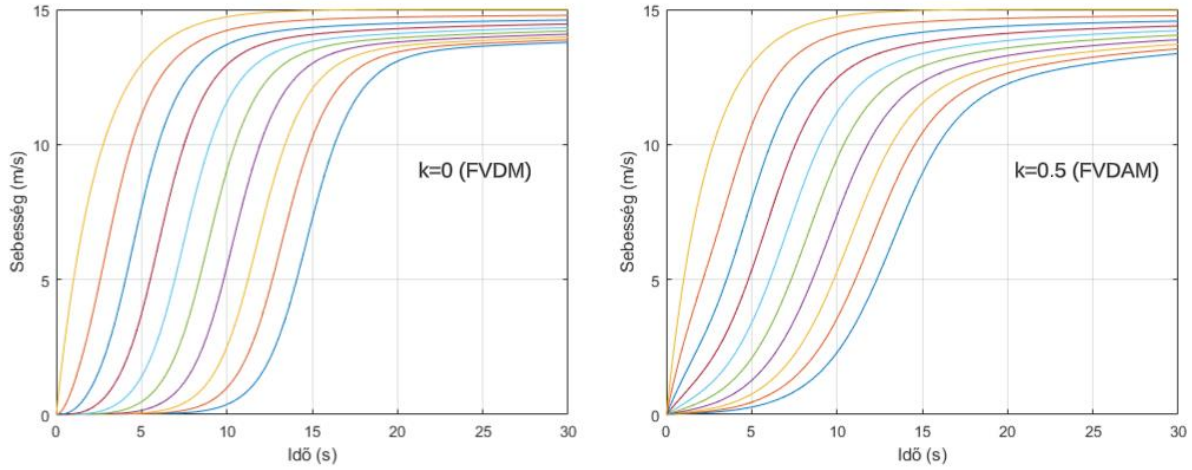
A következő ábrákon a szimulációs eredményeket jelenítjük meg a FVDM és a FVDAM modell esetében. A 2.3.1 -es ábra 10 autó pozícióját szemlélteti idő függvényében a FVDM és a FVDAM modellek esetében, vagyis amikor k=0-át helyettesítünk az (5)-ös egyenlet FVDAM modellbe, akkor megkapjuk a (3)-as egyenletet, ami a FVDM modell. A FVDAM modellek esetében, a cikknek megfelelően k=0.5-re választottuk.



2.3.1. ábra – Autók pozíciója

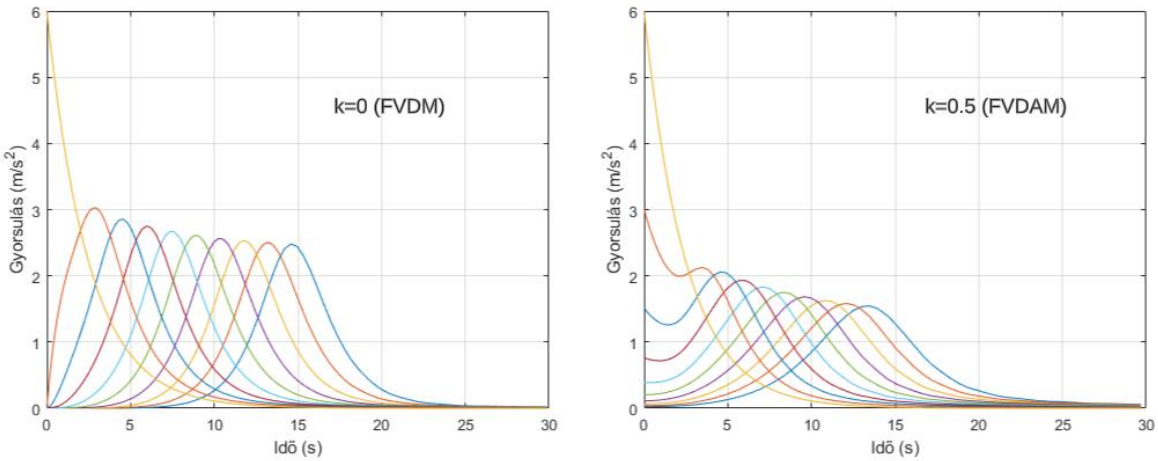
Az ábrán észrevehető, hogy az autók már időben hamarabb változtatják a pozíciójukat, tehát hamarabb reagálnak az előttük haladó autóra. Érdekes megjegyezni, hogy idő függvényében két pozíció közötti távolság egyre növekszik mindkét modell esetében, aminek a hatását látni fogjuk a következő ábrákon.

A következő szimulációs ábrán az autók sebessége van ábrázolva idő függvényében mindkét modellre. Mivel a FVDAM modell figyelembe veszi az előző autó gyorsulását ezért látható, hogy a követő autók esetében hamarabb nő a sebesség és lassulás esetén is hamarabb csökken a sebesség a FVDM modellhez képest. Ez azt eredményezi, hogy a követő autók nem fognak hirtelen nagy sebességre kapcsolni és nem fognak hirtelen nagyot fékezni sem. Itt is észrevehető mindkét modell esetében, hogy az autók sebessége nem áll be egy konstans sebességre, ami megmagyarázza az előbbi ábrán lévő pozíció távolságok növekedését.



2.3.2. ábra – Autók sebessége

A 2.3.3-as ábrán már az autók gyorsulását láthatjuk idő függvényében ábrázolva. A FVDM modellnél a vezető autó gyorsulása $6\,m/s^2$-ből indul és exponenciálisan csökken 0-ba, míg a többi autó gyorsulása $0\,m/s^2$-ből exponenciálisan növekszik egy ideig és utána exponenciálisan csökken 0-ba, ami minden követő autó esetén egy-egy Gauss görbét határoz meg. A FVDAM modellnél megfigyelhetjük, hogy az elején lévő követő autók már egy adott gyorsulással indulnak $\mathrm{t} = 0$ időpillanatban és sokkal jobban követik a vezető autót. A hátrább lévő autók már közel, de nem teljesen $0\,m/s^2$ gyorsulással indulnak és itt is exponenciális növekedést vehetünk észre, viszont itt a Gauss görbék már sokkal laposabbak, mint a FVDM modellnél, amiből arra következtethetünk, hogy a követő autók sokkal jobban követik a vezető autót a FVDAM modell esetében, mint a FVDM modellnél.



2.3.3. ábra – Autók gyorsulása (FVDM)

3. Célkitűzések

A városok forgalmasabb helyein napközben (munkába, iskolába menetkor vagy jövetkor) torlódások alakulnak ki a sok jármű miatt. Ilyenkor az utak áteresztő kapacitása meghaladja a járművek számát. Ezekre a torlódásokra a jelzőlámpák nincsenek megfelelően felkészítve. A jelzőlámpák fő célja az lenne, hogy a torlódásokat enyhítse, megfelelően irányítsa a forgalmat és összességében minél több járművet engedjen át egy adott időegység alatt.

Ezen problémák enyhítésére egy olyan rendszert szeretnénk létrehozni, amely megfelelően irányítja a forgalmat, nagy forgalom esetében is illetve nyugodtabb közlekedési periódusokban is. Egy kereszteződésben ha torlódások alakulnak ki, akkor növelnünk kellene annak a jelzőlámpának a zöld idejét, ahol túl sok autó áll sorban és ezzel egyszerre csökkentenünk a többi jelzőlámpa zöld idejét. Éjszakai forgalom esetén, amikor viszonylag kevés jármű közlekedik akkor az lenne a cél, hogy az autók ne álljanak feleslegesen a jelzőlámpák előtt, ha a kereszteződésben egyetlen jármű sem közlekedik. Ilyenkor a rendszer vegye észre a közeledő járművet és úgy állítsa be a jelzőlámpákat, hogy a közeledő jármű zöld jelzést, összességében zöldhullámot kapjon.

Egy másik fontos célja a dolgozatnak, hogy olyan útvonalakat valósítsunk meg, ahol megfelelő sebességgel haladva az utasok végig zöld jelzéseket kapnak és nem kell várakozniuk a jelzőlámpák előtt. Ezeket az útvonalakat zöldhullámnak nevezzük és nagyon segít a torlódások elkerülésében, illetve a környezetet is kevésbé szennyezik így az autók, mert elkerülik a fölösleges lassítási és gyorsítási szakaszokat. Ennek a megvalósítására szükséges az egymást követő jelzőlámpák összehangolása.

A következő cél, amit a dolgozatunkba szeretnénk megvalósítani az a torlódási szakaszok forgalmának az elkerülése. Ez a cél azt jelenti, hogy amikor egy hosszabb útszakaszon torlódások alakulnak ki és ezt előre látjuk, akkor a torlódások előtt lévő jelzőlámpák úgy tereljék a járműveket, hogy azok egy hosszabb útvonalon hamarabb érjenek a céljukhoz, mint ha a nagy forgalomba vették volna az irányt.

Összefoglalva a célkitűzéseink a következőek:

- torlódások elkerülése és enyhítése
- megfelelő szabályozás nagy és kis forgalom esetén is
- zöldhullám kialakítása
- elkerülő utak ütemezése

4. Forgalomirányítás szimulációban

4.1. Szimulációs szoftver

Közlekedési hálózatok modellezésére és elemzésére a SUMO[4], MATSim[5], VISSIM[6] vagy AIMSUN[7] szoftverek állhatnak rendelkezésünkre. Mi ezek közül a szimulációk közül a SUMO (Simulation of Urban MObility) szoftvert választottuk. Ez egy nyílt forráskódú szimulációs szoftver, mely lehetővé teszi, hogy a felhasználók a saját igényeik,céljaik szerint módosíthassák a szimulációt, valamint lehetővé teszi a saját irányítási algoritmusok beépítését. Nagyon sok lehetséges beépített függvénnyel és részletes leírással, dokumentációval rendelkezik[8]. Ez a szoftver megengedi a felhasználók számára, hogy különböző környezetet: városi, vidéki területeket vagy esetleg autópályák közlekedését szimuláljanak. Nagy előnye ennek a szoftvernek, hogy több kiegészítő eszköz és interfész is kapcsolódik a szimulációhoz. Ezenkívül pedig a szimulációba az OSM[9] (OpenStreetMap) térkép is bele van építve, aminek a segítségével már előre felépített infrastruktúrák vannak előállítva.

A SUMO telepítése után a “tools” nevezetű mappában az “osmWebWizard.py” szkriptet futtatva megjelenik a böngészőben az OpenStreetMap. Itt a felhasználó bármilyen helyre rá tud keresni és könnyedén tud generálni egy-egy szcenáriót, amit a szkript importál SUMO szimulációba. Még generálás előtt különböző paramétereket lehet beállítani, mint például: szimuláció időtartamát, buszok, teherautók, gyalogosok, vonatok, hajók szimulálását, autók sűrűségét, jobb vagy bal oldali közlekedésmódot stb. A “Generate Scenario” gombra kattintva kigenerálja számunkra a szimulációhoz szükséges fájlokat és elindítja a szimulációs környezetet, ahol már szimulálni tudjuk az alapértelmezett beállításokkal a kiválasztott területet. Ilyen például a program indítását követő Berlin központját ábrázoló térkép.

[4] https://eclipse.dev/sumo/

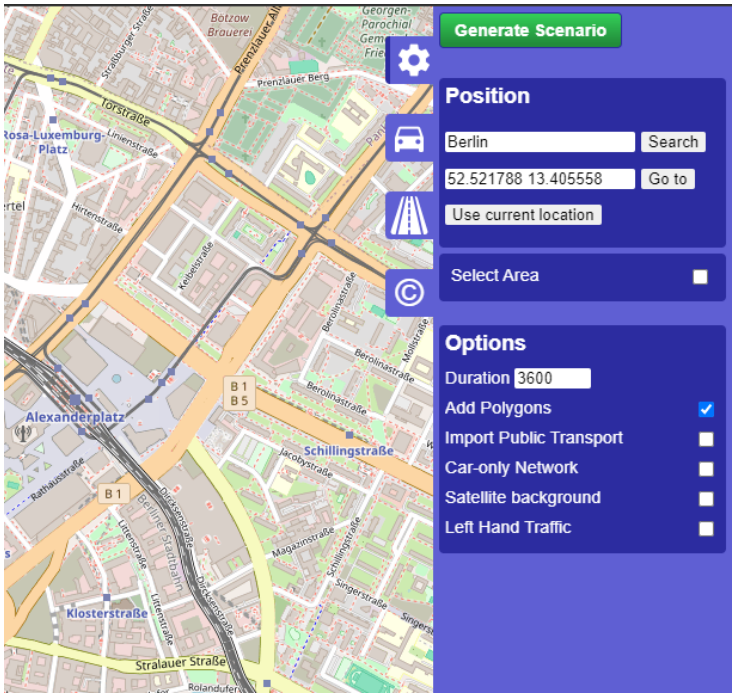
[5] https://matsim.org/

[6] https://www.ptvgroup.com/en/products/ptv-vissim

[7] https://www.aimsun.com/

[8] https://sumo.dlr.de/docs/index.html

[9] https://www.openstreetmap.org/



4.1.1. ábra – Szcenárió generálás OSM térképen

Az egyik legfontosabb generált fájl a “*.sumocfg”-vel kiterjesztett fájl, amiben a szimuláció konfigurációja, beállítása van leírva. Ebbe a konfigurációs fájlban kell megadni a bemeneti “*.xml” fájlokat, amik tartalmaznak információkat a közlekedést szimuláló gráf éleiről, csomópontjairól és ezek kapcsolatairól, valamint útvonalakat és utazási információkat minden autóról. Kiegészítő fájlként alapjáraton egy fájl adott, ami az épületek méretét és dimenzióját tárolja. Ezek után különböző feldolgozási, újratervezési beállításokat lehet megadni, naplózási beállításokat és a grafikus felhasználói felülettel kapcsolatos dolgokat lehet beállítani. Egy generált konfigurációs fájl a függelékben látható módon néz ki. A rendezettebb munka érdekében az “*.xml” fájlokat egy mappába helyeztük, majd az elérési útvonalakat ennek megfelelően megváltoztattuk a konfigurációs fájlban is.

4.2. TraCI

A TraCI (Traffic Control Interface) egy olyan protokollt biztosít, amely lehetővé teszi, hogy a külső alkalmazások kommunikáljanak a SUMO-val. A TraCI segítségével egy kapcsolatot tudunk létrehozni a SUMO és a Python szkriptek között, ahol adatokat tudunk lekérdezni, beállítani, de akár különböző szabályozásokat vagy gépi tanulásokat is lehet implementálni.

Python-ban a TRACI interfész használatához szükségünk van a “traci” könyvtárra. Ennek a könyvtárnak a segítségével tudunk parancsokat kiadni a szimuláció futtatására, a járművek vezérlésére: sebesség, gyorsulás lekérdezése, beállítása, jelzőlámpák adatainak lekérdezése és beállítása, sávterület detektorok beállítása és lekérdezése stb. A szimuláció indítását első lépésben a következőképpen tudjuk a legegyszerűbben lekódolni:

```
import traci

sumoCmd = ["sumo-gui", "-c", "osm.sumocfg"]
traci.start(sumoCmd)

step=0
while traci.simulation.getMinExpectedNumber()>0:
        traci.simulationStep()
        step += 1
traci.close()
```

4.2.1. ábra – SUMO szimuláció indítása Python környezetben

Ebben a kódrészletben importáljuk a “traci” könyvtárat, majd a sumoCmd listában megadjuk a szimuláció indításához szükséges parancsokat. A “sumo-gui” paranccsal megadjuk, hogy egy grafikus felhasználói felületet indítson (GUI), a “-c” argumentum arra utal, hogy egy konfigurációs fájlt adunk meg a SUMO-nak, majd végül megadjuk a SUMO konfigurációs fájlt, ami a mi esetünkben az “osm.sumocfg”. Ezt a konfigurációs fájlt a 4.1.2-es ábra szemlélteti.

A “traci.start(sumoCmd)” parancs elindítja a szimulációt a megadott konfigurációs fájlal és létrehozza a TraCI kapcsolatot. Ezek után egy step változót 0 kezdeti értékkel létrehozunk, majd egy while ciklust indítunk, ami addig fut, míg a várhatóan jelen lévő járművek száma 0-nál nagyobb. A “traci.simulationStep()” parancs minden iterációban egy szimulációs lépéssel lépteti a szimulációt és közben mi is növeljük a step változót, amivel nyomon követjük a szimulációs lépést. Végül pedig a “traci.close()” paranccsal megszakítjuk a Traci kapcsolatot és lezárjuk a szimulációt.

A jelzőlámpák megfelelő szabályozásához szükségünk van a jelzőlámpáknál kialakult sorok hosszára. Ezekre szolgálnak a sávterület detektorok vagy angol megnevezéssel a “lane area detector”-ok. Minden egyes sávra külön detektorokat kell megadjunk. Ezeket a detektorokat egy “*. xml” fájlban adjuk meg, amit “additional-file”-ként hozzáadunk a konfigurációs fájlhoz is.

A 4.2.2-es ábrán egy detektor létrehozását láthajuk, ami egy megadott hosszúságú sávon figyeli a járművek számát. A detektor létrehozásához szükséges megadnunk egy azonosítót (id-t), a megfigyelt sávok azonosítójait(egymás után lévő sávokat össze lehet kötni egy detektorba),

a sáv kiinduló és végpontját, ameddig szeretnénk, hogy tartson a detektálás egy kimeneti fájlt és a jelzőlámpa azonosítóját.

```
<?xml version="1.0" encoding="UTF-8"?>
<additional>
    <laneAreaDetector   id="laneAreaDetector1"   lanes="-948993200#1_0"   pos="0"
endPos="39.7"                                                    file="output.xlsx"
tl="cluster_1936414352_1936414379_26003429_7516041220_#2more"/>
 </additional>
```

4.2.2. ábra – Sávterület detektor létrehozása

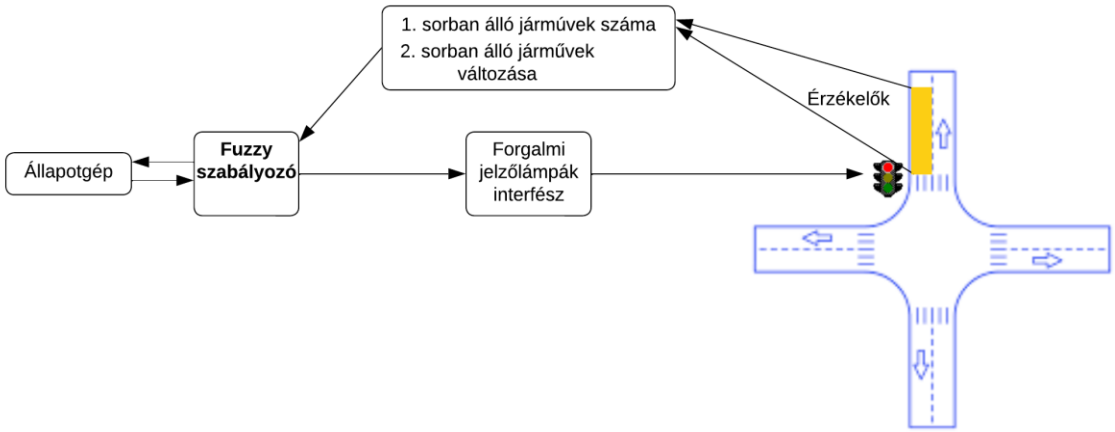
4.3. A városi forgalom irányításának intelligens tömbvázlata

Valós városi forgalomirányításra két hagyományos módszert alkalmaznak alapvetően. Az egyik típus előre beállított ciklusidőt használ a lámpák váltásához. A másik típus kombinálja az előre beállított ciklusidőt a közelségérzékelőkkel, amelyek aktiválhatják a ciklusidő vagy a lámpák váltását. Kevésbé forgalmas utcák esetében, amelyek nem igényelnek rendszeres zöld lámpa ciklust, a közelségérzékelők aktiválják a lámpa váltását, amikor autók jelennek meg. Ez a fajta irányítás előzetes ismereteket igényel az adott kereszteződés forgalmi viszonyairól és szükséges a jelző ciklusidőt és a közelségérzékelőt az adott kereszteződéshez testre szabni. [9]

A fuzzy szabályozással működő forgalomirányító rendszer alternatívája a hagyományos forgalomirányító rendszereknek. A fuzzy szabályozással irányított rendszerekben érzékelőket használnak, amik segítségével megszámolják az autókat, míg a közelségérzékelőkkel ellátott rendszerek csak az autók jelenlétét jelzik. Így ezek a rendszerek képesek meghatározni a forgalom sűrűségét és ahogy a forgalmi áramlások kezdenek ingadozni, úgy a fuzzy irányítás alkalmazkodik a megfelelő forgalmi viszonyokhoz és azoknak megfelelően változtatja a jelzőlámpákat. [9]

A valóságban a fuzzy szabályozással működő forgalomirányító rendszereknél minden sávban két elektromágneses érzékelőt helyeznek el. Az első érzékelő a jelzőlámpák mögött áthaladó autókat számolja, míg a második érzékelő, amely az első érzékelő mögött található, az útkereszteződéshez közeledő autók számát számolja a jelzőlámpáktól egy meghatározott távolságra. A 4.3.1-es ábrán sárga színnel jelöltük a két érzékelő közti részt és bemutattuk a fuzzy szabályozás tömbvázlatát. A jelzőlámpáknál sorban álló autók számát az érzékelők által leolvasott értékek különbségéből határozzák meg. Ez ellentétben áll a hagyományos irányító rendszerekkel, amelyek közelségérzékelőt helyeznek el minden forgalomlámpa előtt, és csak a kereszteződésnél várakozó autó jelenlétét érzékelik, nem pedig a várakozó autók számát. A

fuzzy szabályozó felelős a a zöld jelzés hosszának szabályozásáért a forgalmi viszonyoknak megfelelően. [9]



4.3.1. ábra – Egy kereszteződésben lévő jelzőlámpák fuzzy szabályozásának tömbvázlata

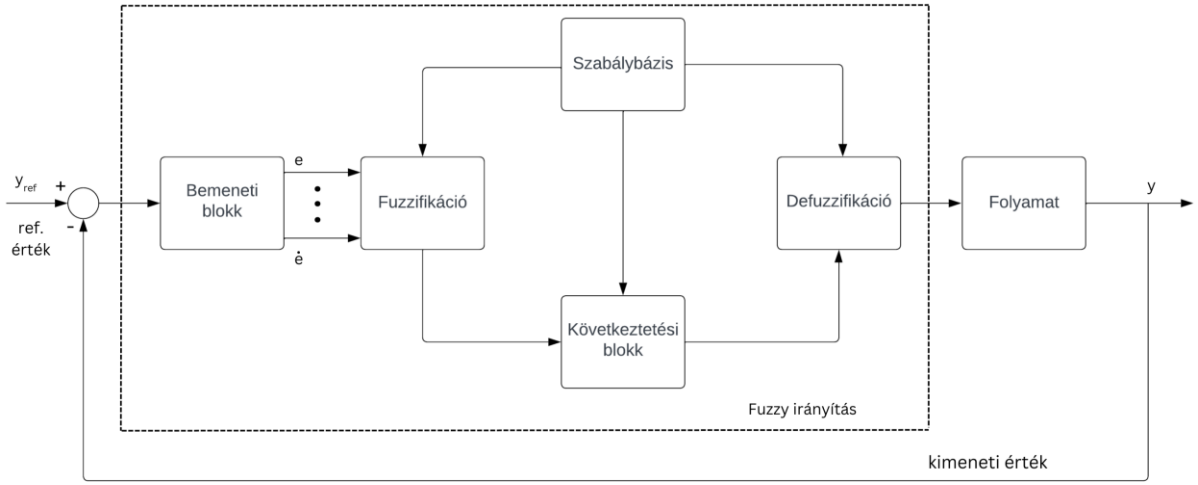
4.4. Fuzzy irányítás

A Fuzzy logikát 1965-ben Lotfi Zadeh matematikus mutatta be. A Fuzzy logika egy matematikai eszköz a bizonytalanságok kezelésére. A lágy számítástechnika területén fontos szerepet játszik a szavakkal való számítás fogalmának bevezetésével. Lehetőséget biztosít a nyelvi konstrukciók reprezentálására. Nyelvi konstrukciók lehetnek a következőek: nagyon magas, sok, közepes, alacsony, kevés, gyakran, ritkán stb. Általánosan a Fuzzy logika olyan következtetéseket biztosít, ami az emberi gondolkodásmód képességeit alkalmazza. A hagyományos bináris halmazelmélet éles eseményeket ír le, amelyek vagy bekövetkeznek, vagy nem. Ezzel szemben a Fuzzy halmazok képesek modellezni a bizonytalan vagy kétértelmű adatokat. [10]

Tan Kok Khiang és társai cikkében [9], valamint Javed Alam és társai cikkében [10], egy kereszteződés irányítását valósították meg Fuzzy irányítással. Ők úgy választották meg a kereszteződést, hogy négy irányból érkezhetnek és csak előre haladhatnak a járművek, vagyis nem térhetnek el jobbra vagy balra a kereszteződésben. A négy sáv iránya megfelel Észak, Dél, Nyugat és Keletnek. Ha az északi és déli oldalon zöld a lámpa, akkor ezt érkező oldalnak tekintjük, míg a nyugati és keleti oldalt várakozó oldalnak tekintjük, és fordítva. A Fuzzy irányítást úgy oldották meg, hogy a egyik bemenetnek megadják a várakozó oldalon sorban álló autók hosszát és a másik bemenetnek az érkező autók hosszát, majd ezeket megfelelően

fuzzifikálva és a szabályrendszer alapján olyan következtetést hoznak, ami a kimeneten megadja a zöld idő meghosszabbításának idejét. Ez a meghosszabbítási idő az aktuális zöld lámpára vonatkozik, ami egy pozitív érték abban az esetben ha meghosszabbításra van szükség, de lehet 0 érték is abban az esetben, ha nincs szükség hosszabbításra. Ezekre a cikkekre alapozva döntöttünk mi is a Fuzzy irányítás mellett. Nálunk a Fuzzy irányítás abban tér el a cikkekben említettől, hogy mi a zöld jelzést kapott sávokon sorban álló járművek hosszát vesszük egyik bemenetnek és a második bemenetnek a zöld jelzést kapott sávokon sorban álló járművek hosszának a változását vesszük. Mi esetünkben a kereszteződésekben a térképnek megfelelően balra vagy jobbra is lehet haladni. A Fuzzy kimenete nálunk az aktuális zöld idő befolyásolása: ha csökkenteni szeretnénk akkor egy negatív érték, ha növelni akkor egy pozítiv érték vagy 0 ellenkező esetben.

A Fuzzy irányítás a Fuzzy logika egy gyakorlati alkalmazása, amelyet különösen olyan rendszerek esetében alkalmaznak, ahol a hagyományos matematikai modellezés nehézkes vagy lehetetlen. Ez a módszer az emberi gondolkodás és döntéshozatal egyes aspektusait modellezi. Az irányítást nemlineáris rendszerekre alkalmazzák, ahol bizonytalanságok vagy pontatlan információk vannak és a megfelelő beállításokkal a Fuzzy könnyen és hatékonyan képes irányítani a rendszert. A Fuzzy irányítás lépéseit a 4.4.1-es ábrán láthatjuk.



4.4.1. ábra – Fuzzy irányítás lépései

A fuzzy irányításnak egy vagy akár több bemeneti adatot is megadhatunk, amiből a lépések során egy kimeneti értéket térít vissza. Első lépésként a bemeneti változókat fuzzy halmazokká alakítja. Ez azt jelenti, hogy minden bemeneti adatot hozzárendel egy vagy több tagsági függvényhez és ezek a függvények meghatározzák a bemeneti értékekhez tartozó tagsági

fokokat. A tagsági függvények sokfélék lehetnek, de a leggyakrabban használtak a háromszög és a trapéz alakú tagsági függvények.

A szabálybázisban megadjuk a szabályokat ami alapján szeretnénk, hogy döntsön a rendszer. Ezek a szabályok emberi gondolkodásmódon alapulnak. Például ha jelzőlámpáknál sorban álló autók hossza az egyik bemenet: “hosszú” és a másik bemenet a változása: “növekszik”, akkor a fuzzifikált kimenetünk a zöld idő “növelése” lehet. Ez a szabályrendszer ha-akkor (if-then) alakban van felépítve.

A következtetés során a szabálybázis segítéségével megkapjuk a fuzzifikált kimenetet és annak a mértékét. A fuzzifikált kimenetek ugyanúgy tagsági függvények, de egyszerűbb megvalósításban szingletonok is lehetnek.

A defuzzifikáció az a lépés amikor fuzzifikált kimeneteket defuzzifikáljuk és megkapjuk a pontos (crisp) kimeneti értéket. A defuzzifikálásra többféle módszer is létezik, ezek közül a legelterjedtebb a súlypont módszer, a területközéppont módszer és a területfelezéses módszer.

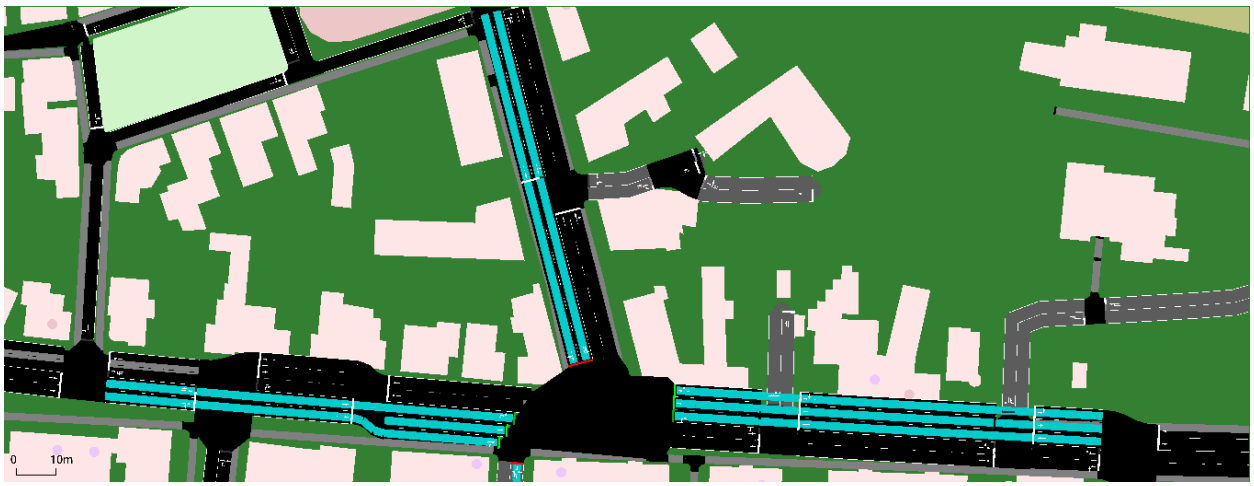
Első lépésben egy kereszteződés irányítását oldottuk meg fuzzy szabályozóval. Ezt úgy valósítottuk meg, hogy felosztottuk a kereszteződést három vagy négy szekvenciára a kereszteződés bonyolultságának megfelelően. Egy szekvencia alatt azokat a sávokat értjük, akik egyszerre kapnak zöld jelzést és a sávokban lévő autók a forgalmi szabályoknak megfelelően tudnak haladni egymás útvonalait nem keresztezve. Ha a 4.4.2-es ábrát tekintjük, ami a “Fortuna” buszmegálló melletti kereszteződés Marosvásárhelyen, akkor ezt a kereszteződést négy szekvenciára bonthatjuk fel:

1. nyugatról és keletről közlekedő járművek kapnak előre és jobbra zöld jelzést
2. nyugatról és északról haladó járművek kapnak balra zöld jelzést és az északi és déli oldalról érkező járművek is kapnak jobbra zöld jelzést
3. északról és délről közlekedő járművek kapnak előre és jobbra zöld jelzést
4. északról és délről közlekedő járművek kapnak balra zöld jelzést és a keleti és nyugati oldalról érkező járművek is kapnak jobbra zöld jelzést

Ezek után minden szekvenciának kezdetben megadtunk egy időtartamot, úgy hogy az elején minden szekvencia időtartama egyenlő legyen. Így az irányítás nem a forgalomnak megfelelő szekvencia időtartamokkal indul, de ezt az irányítással csökkentjük vagy növeljük a forgalomnak megfelelően. Ezáltal kialakulnak a forgalomnak megfelelő szekvenciák hosszának az időtartama, ami képes adaptálódni az aktuális forgalomhoz.

4.4.2. ábra – Marosvásárhelyi kereszteződés felosztása szekvenciákra

A teljes Fuzzy irányítást mi építettük fel egy Python szkriptben. Úgy valósítottuk meg, hogy két bemeneti értéket kell megadjuk: az aktuális szekvenciában lévő várakozó járművek számát és ezeknek a változását. Kimenetként pedig visszatérít egy értéket, ami megmondja, hogy hány másodperccel növeljük vagy csökkentsük az adott szekvencia zöld idejét. Ha növelni kell az adott szekvencia zöld idejét akkor a többi szekvenciától egyenlően elosztva levonja azt az időt, ha pedig csökkenteni kell az zöld időt akkor megfelelően elosztva hozzáadja azt az időt a többi szekvencia zöld idejéhez.

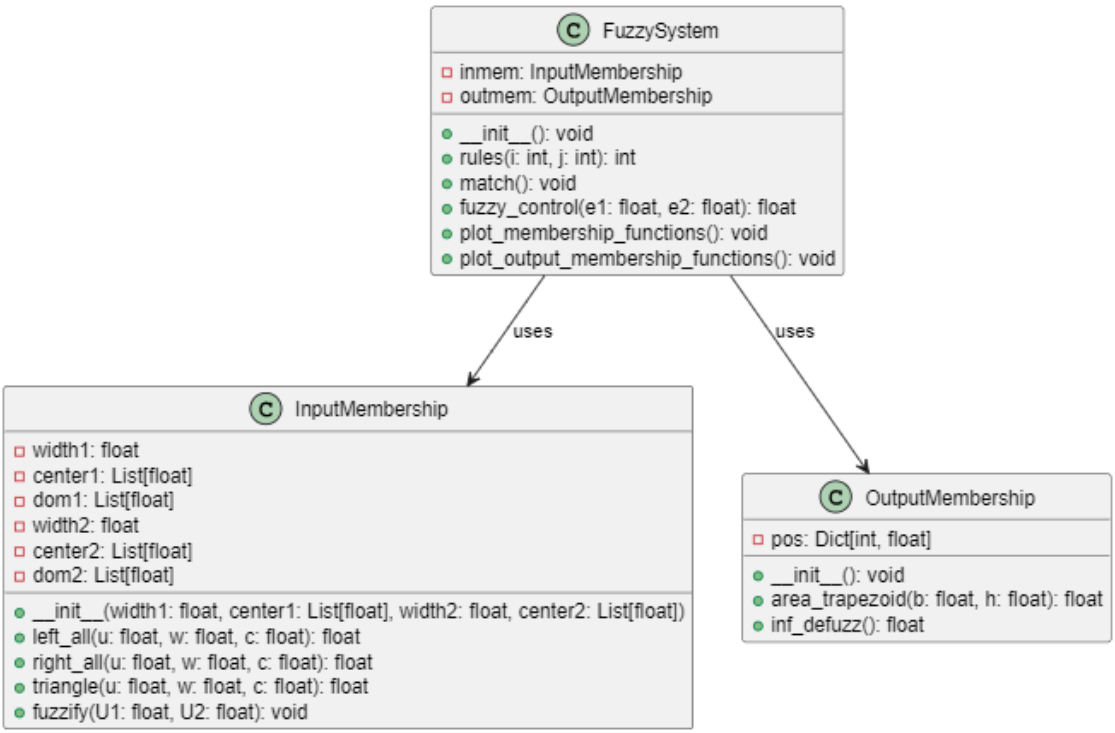


4.4.3. ábra – Sávterület detektorok a szimulációban

Ahhoz hogy lekérdezzük a jelzőlámpáknál sorban álló autókat szükséges minden jelzőlámpa előtti sávra detektorokat beépítsünk. Ezeket a 4.2.2-es ábrának megfelelő sablonnal

hozzuk létre és a szimulációban egy kék réteg tevődik a sávokra, ahogyan ezt a 4.4.3-as ábrán is láthatjuk. Ezen az ábrán egy másik kereszteződés látható Marosvásárhelyről, amit már a szimulációban van. Ezekről a detektorral ellátott sávokról pedig Pythonban a “traci” könyvtáron keresztül le tudjuk kérdezni a detektált járműveket, amik elengedhetetlenül fontosak az irányítás céljából.

A fuzzy irányítást mi egy Python szkriptbe objektum orientált programozással valósítottuk meg. Ez a szkript három osztályt tartalmaz: “FuzzySystem”, “InputMembership” és “OutputMembership”. A kódrészlet osztálydiagramját a következő ábrán szemléltetjük.



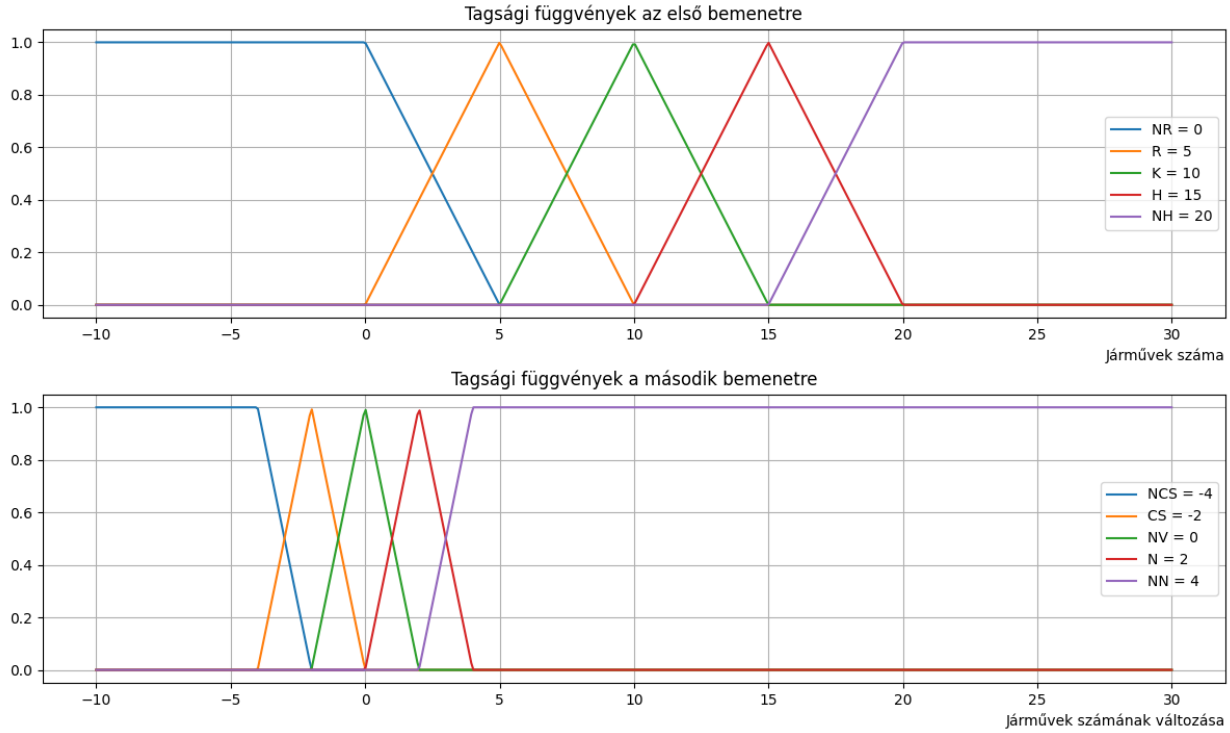
4.4.4. ábra – Fuzzy irányítás osztálydiagramja

Az “InputMembership” osztály a bemeneti értékek fuzzifikációjával foglalkozik. Hat attribútuma van, ami a következőket jelöli:

1. “width1”: egy érték, ami az első bemenet tagsági függvényeinek szélessége
2. “center1”:egy vektor, ami az első bemenet tagsági függvényeinek középpontjait tartalmazza
3. “dom1”: egy vektor, ami az első bemenet tagsági függvényekhez tartozás mértékeit adja meg
4. “width2”: egy érték, ami a második bemenet tagsági függvényeinek szélessége
5. “center2”:egy vektor, ami a második bemenet tagsági függvényeinek középpontjait tartalmazza

6. “dom2”: egy vektor, ami az második bemenet tagsági függvényekhez tartozás mértékeit adja meg.

Ennek az osztálynak egy konstruktora van, ami inicializálja az említett attribútumokat. Ezen kívül pedig még négy metódusa van. A “left_all()”, “right_all()” olyan metódusok, amik a fuzzifikáció során a bemenetekhez hozzárendelik a szélső tagsági függvényekhez való tartozás mértékét. Ezek a tagsági függvények egy-egy fél trapéznak felelnek meg és a bal, illetve a jobb szélen helyezkednek el. A “triangle()” metódus a háromszög tagsági függvényekhez való tartozás mértékét téríti vissza. Ezeknek a metódusoknak a visszatérített értékeit tároljuk a “dom1” és “dom2” vektorokban. Mindkét bemenetre öt tagsági függvényt határoztunk meg és ezt a 4.4.5-ös ábrán tüntettük fel.



4.4.5. ábra – Bemeneti tagsági függvények ábrázolása

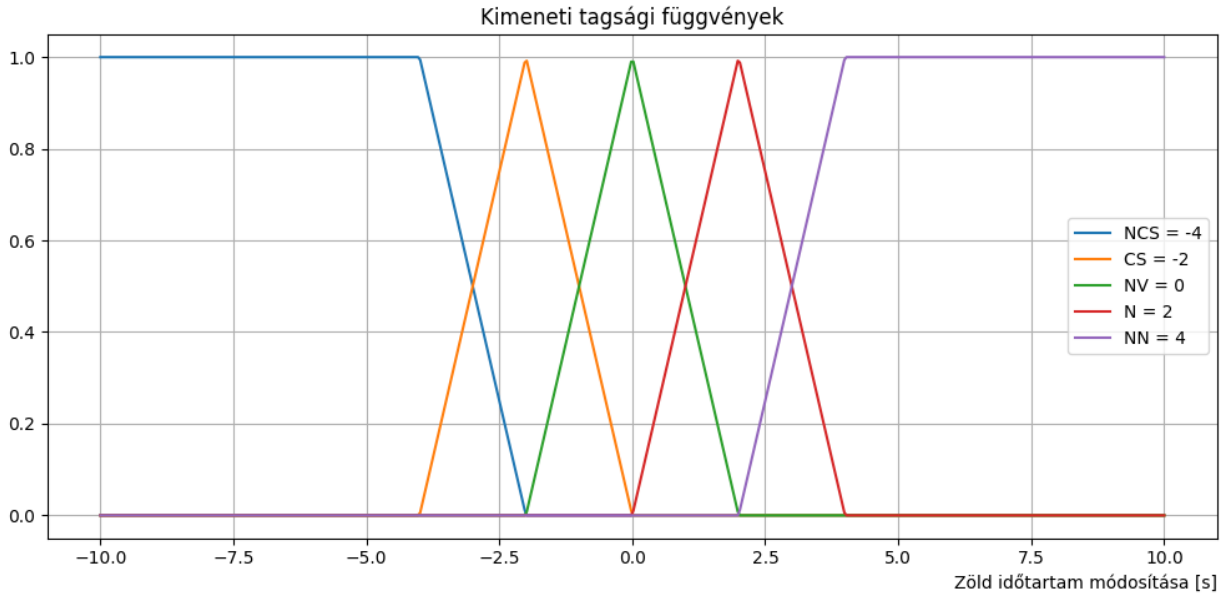
Az első bemenet tagsági függvényei vagy másképp mondva bemeneti változói a szekvencia hosszára utalnak. A tagsági függvények középpontjai pedig a **0**, **5**, **10**, **15** és **20** -ban helyezkednek el. A második bemeneti tagsági függvények a szekvencia hosszának a változását jelöli és középpontjai pedig a **-4**, **-2**, **0**, **2** és **4** -ben helyezkednek el. A bemeneti és a kimeneti tagsági függvények rövidítéseit a 4.4.1-es táblázatban szemléltetjük.

4.4.1. táblázat – Fuzzy tagsági függvények rövidítései

Első bemeneti tagsági függvények		Második bemeneti tagsági függvények		Kimeneti tagsági függvények	
Nagyon **R**övid	**NR**	**NCS**	Nagyon **C**sökken	**NCS**	Nagyon **C**sökkentjük
Rövid	**R**	**CS**	**C**sökken	**CS**	**C**sökkentjük
Közepes	**K**	**NV**	**N**em **V**áltozik	**NV**	**N**em **V**áltoztatjuk
Hosszú	**H**	**N**	**N**övekszik	**N**	**N**öveljük
Nagyon **H**osszú	**NH**	**NN**	Nagyon **N**övekszik	**NN**	Nagyon **N**öveljük

A “fuzzify()” az “InputMembership” osztály negyedik metódusa az előbbi három metódus segítségével a bemeneti változókat fuzzifikálja vagyis fuzzy halmazokká alakítja.

Az “OutputMembership” osztály a kimeneti tagsági függvények defuzzifikációjával foglalkozó osztály, aminek egy “pos” azonosítójú attribútuma van. Ez az attribútum egy szótár (dictionary) típusú adat, amiben kulcs-érték párokat tárolunk. Ez esetben a kulcsok a kimeneti tagsági függvényeket jelölik: **NCS**, **CS**, **NV**, **N**, **NN**. Az értékek pedig a tagsági függvényekhez való tartozást jelölik. A kódrészletben kulcsoknak mi már konkrét értékeket adtunk meg a “NCS”, “CS”, “NV”, “N”, “NN” jelölések helyett, amik a tagsági függvények középpontjai: {-4, -2, 0, 2, 4}. Tehát ha nagyon csökkentjük akkor -4 másodperc és a nagyon növelés esetén 4 másodperc a tagsági függvény értéke. A 4.4.6-os ábrán láthatjuk a kimeneti tagsági függvények ábrázolását.



4.4.6. ábra – Kimeneti tagsági függvények ábrázolása

Az osztálynak egy konstruktora van, ami inicializálja az osztály attribútumát kezdetben 0 értékekkel. Ezen kívül még két metódussal rendelkezik: az “area_trapezoid()” metódus egy “@staticmethod” dekorátorral ellátott függvény, ami nem függ az osztály példányaitól vagy az osztály belső állapotától, a szerepe az, hogy a levágott háromszögből lett trapéz területét kiszámítsa, úgy hogy adott a trapéz nagy alapja és a magassága; az “inf_defuzz()” metódus pedig a “pos” attribútumot végigiterálva a területközéppont módszerrel meghatározza a defuzzifikált kimenetet.

A harmadik osztály: “FuzzySystem” a szabálybázis és következtetésekkel foglalkozó osztály. Két attribútummal rendelkezik:

1. “inmem”: egy “InputMembership” objektumot tárol, tehát a bemeneti osztály példányát
2. “outmem”: egy “OutputMembership” objektumot tárol.

Az osztálynak hat metódusa van, amit a következőkben röviden bemutatunk. A __init__() konstruktor inicializálja az attribútumokat, vagyis létrehoz egy “InputMembership” és egy OutputMembership” objektumot. A “rules()” metódus úgyszintén független az osztály példányától tehát ez is egy “@staticmethod” kulcsszóval van ellátva a megfelelő működés érdekében. Ez a metódus a szabályokat definiálja. Két paramétere van: egyik a szekvencia hossza (sorban álló járművek száma) és a másik a szekvencia hosszának a változása. A szabályok alapján visszatérít egy értéket, ami megfelel az egyik fent említett kimeneti tagsági függvénynek.

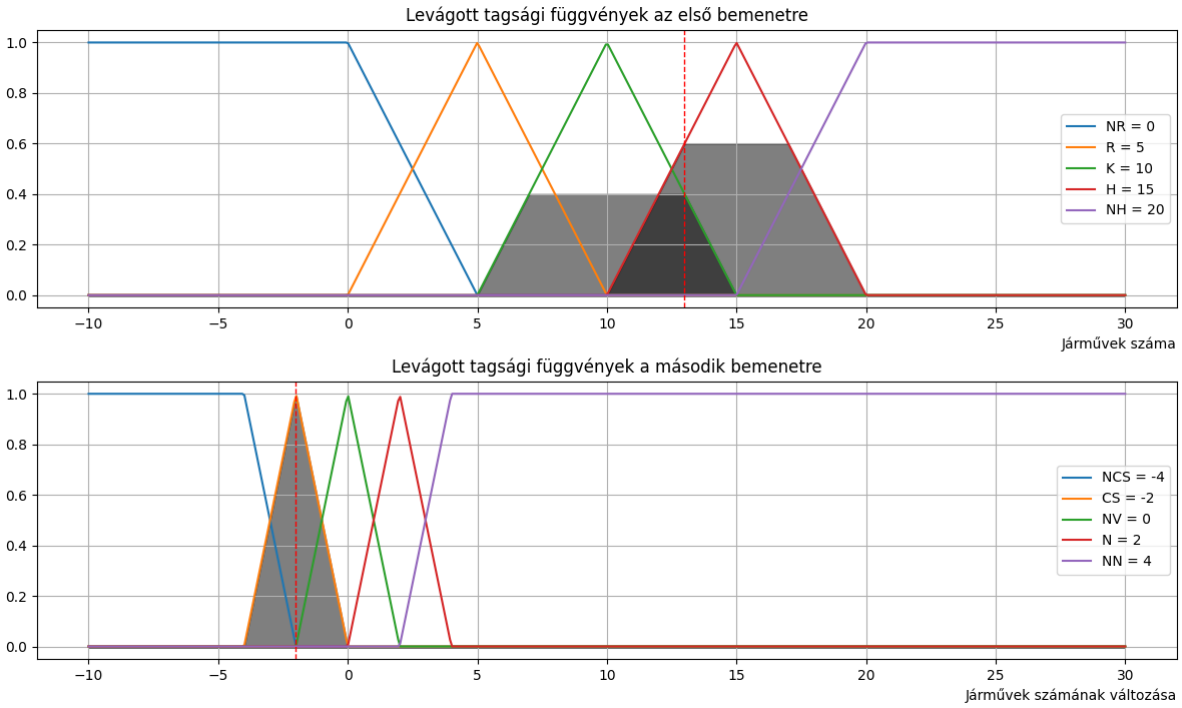
4.4.2. táblázat – Fuzzy szabályok

	NR	**R**	**K**	**H**	**NH**
NCS	NCS	NCS	CS	NV	N
CS	NCS	CS	NV	N	N
NV	CS	NV	N	N	NN
N	NV	NV	N	NN	NN
NN	N	N	NN	NN	NN

A szabályokat az 4.4.2-es táblázatban adtuk meg, ahol a táblázat első sora az első bemenet tagsági függvényeinek rövidítéseit tartalmazza, az első oszlop pedig a második bemenet tagsági függvényeinek rövidítését tartalmazza. A táblázat többi cellájában pedig a kimeneti tagsági függvények rövidítései láthatóak. Ha ki szeretnénk olvasni néhány értéket a táblázatból akkor a következőképpen tehetjük meg:

- 2. sor 2. oszlop: Ha a szekvencia hossza nagyon rövid (NR) és a változása nagyon csökken (NCS) akkor a nagyon csökkentjük (NCS) a zöld időt
- sor 4. oszlop: Ha a szekvencia hossza közepes (K) és a változása növekszik (N) akkor növeljük (N) a zöld jelzést.

A fuzzy szabályozás következtetési mechanizmusa hasonlít az emberi gondolkodási folyamatra. Ez az a pont, ahol fuzzy szabályozás technológiája a mesterséges intelligenciával társul. Az emberek tudattalanul használnak szabályokat a cselekedeteik végrehajtásában. Például egy közlekedést irányító rendőr, aki egy 4 irányú kereszteződést irányít nagyjából a következőképpen gondolkozik: ha északról és délről nagyobb a forgalom, mint a nyugati és keleti irányban, akkor az északról és délről érkező forgalmat hosszabb ideig engedélyezi, mint a nyugati és keleti oldalról érkezőt. [9]

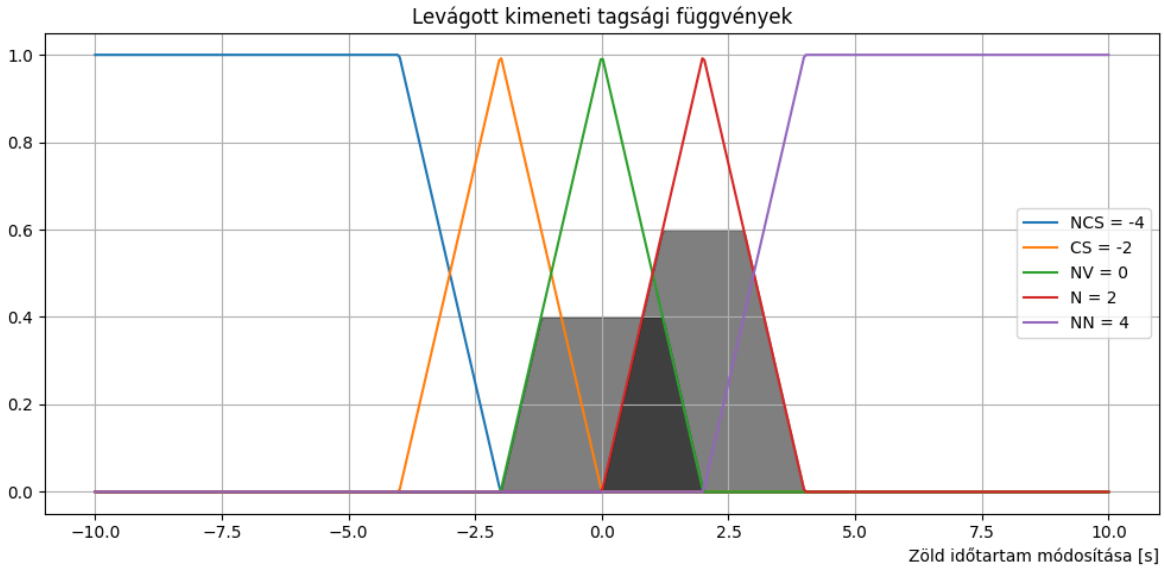


4.4.7. ábra – Bemeneti tagsági függvények levágása

A “match()” metódus az “InputMembership” osztály “dom1” és “dom2” vektorain két for ciklussal végig iterál. Ha mindkét vektorban az aktuális elem nem nulla akkor azokra a tagsági

függvényekre meghívjuk a “rules()” metódust, ami visszatéríti számunkra a kimeneti tagsági függvényt. A kimeneti tagsági függvényhez való tartozás mértékét a két bemeneti tagsági függvényhez tartozás mértékének a minimuma adja meg, vagyis az aktuális “dom1” és “dom2” vektor értékeinek a minimuma. Ezeket az értékeket eltároljuk az “OutputMemory” osztály “pos” attribútumában. Ha az adott szabály többször is létezik akkor mindig a legnagyobb (max) mértékét tároljuk el a “pos” attribútumban. Ezt a technikát nevezik MiniMax technikának a fuzzy következtetések esetében. Erre egy példát láthatunk a 4.4.7. és 4.4.8-as ábrán.

Első bemenet: “in1=13” értéket adtunk meg. Ez a bemeneti érték két tagsági függvényhez is hozzátartozik: “K”-hoz 0.4 mértékkel és “H”-hoz 0.6-os mértékkel. A második bemenet: “in2=-2” értéket kapta, ami már csak egy tagsági függvényhez tartozik, a “CS”-hez, 1-es mértékkel, ami teljes hozzátartozást jelent. A szabályoknak megfelelően pedig két kimeneti tagsági függvény fog élni, amit a fent említett bemeneti tagsági függvények mértékének a minimuma határozza meg. Ezt a 4.3.8-as ábra szemlélteti. A “K” és a “CS” tagsági függvényekből a szabályok alapján megkapjuk az “NV” kimeneti tagsági függvényt 0.4-es mértékkel, a “H” és a “CS” tagsági függvényekből pedig az “N” kimeneti tagsági függvényt 0.6-os hozzátartozással. A defuzzifikálás során pedig a levágott háromszögekből létrejött trapézokból a területközéppont módszer segítségével határozzuk meg a kimenetet. Ezeket a levágott területeket szürkén szemléltettük az ábrán.



4.4.8. ábra – Levágott kimeneti tagsági függvények

A következő metódus a “fuzzy_control()”, ami a paraméterként kapott két bemeneti értékre meghívja a “fuzzify()” metódust. A fuzzifikáció után meghívja az előbb említett “match()” metódust és ezek után meghívja az “inf_defuzz()” metódust, amit vissza is térít. Ezen kívül még a könnyebb átláthatóság tekintetében beépítettünk egy “plot_input_membership_functions()” metódust, ami ábrázolja a bemeneti tagsági függvényeket és egy “plot_output_membership_functions()” metódust, ami a kimeneti tagsági függvényeket ábrázolja.

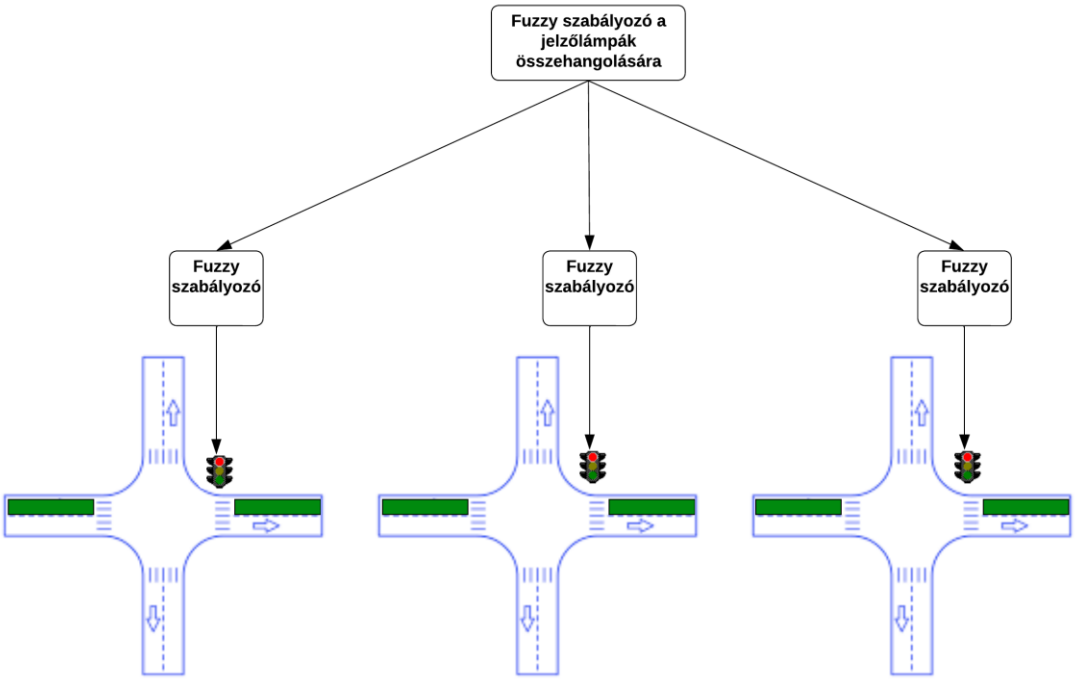
4.5. Szomszédos kereszteződésben lévő jelzőlámpák összehangolása

Az urbanizált közúti forgalom növekedésével és az úthálózat sűrűségének növekedésével a szomszédos úti kereszteződések közötti összefüggés egyre nyilvánvalóbbá válik. Egy kereszteződés forgalmi jelzőlámpájának beállítása hajlamos befolyásolni több szomszédos kereszteződés közúti forgalmát. Ennek a torlódásnak az idő múlásával fokozatosan terjedhet néhány háztömbnyire és a kereszteződés körüli régiókra. Ezért az egyre növekvő városi forgalmi jelzőlámpa vezérlés igényei egyre több fejlett vezérlési technológia integrációját követelik meg, hogy egy vezérlési paraméter beállításával elérhessék a városi szintű forgalom dinamikus koordinációját, kielégítve a forgalmi igényeket, és új egyensúlyt teremtsenek az úti forgalom és a közlekedési igények között. [11]

Yi Zhang és társa: Gengsheng Huang cikkében [11] a szomszédos kereszteződésekben lévő forgalmi jelzőlámpák összehangolását valósítják meg fuzzy szabályozással, így egy zöldhullámot effektust hozva létre egy adott irányban. Ennek a cikknek a segítségével mi is megvalósítottuk a szomszédos kereszteződések jelzőlámpáinak szinkronizálását. A kereszteződéseket külön-külön egy-egy fuzzy szabályozó irányít, ami a forgalomnak megfelelően nagy mértékben befolyásolja a zöld jelzések időtartamát. A zöldhullám effektus kialakításához még szükségünk van egy fuzzy szabályozóra, ami kisebb mértékben befolyásolja a zöld jelzések időtartamát, de egy viszonylag hosszabb idő elteltével képes a jelzőlámpák összehangolására annak érdekében, hogy zöldhullám alakuljon ki a kívánt irányba. A 4.5.1. ábra bemutatja a jelzőlámpák összehangolásának tömbvázlatát, ahol zöld színnel jelöltük azokat az útszakaszokat, ahol a zöldhullám effektust hozzuk létre.

A fennebb említett cikk [11] alapján építettük fel a magasabb szinten lévő fuzzy szabályozót, ami összehangolja a szomszédos jelzőlámpákat. Ezt a fuzzy szabályozót a 4.4.4-es ábra osztálydiagrammjának mintájára építettük fel annyi különbséggel, hogy a tagsági függvényeknek nem egyforma, hanem különböző szélességeket adtunk meg. A szabályozó első

bemenete az aktuális kereszteződésben lévő λ+2 szekvenciában a sorban álló járművek hosszának és a λ+1 szekvenciában sorban álló járművek hosszának a különbsége. A második bemenet pedig a szomszédos kereszteződésbe beérkező járművek számának és az onnan kimenő járművek számának a különbsége, vagyis a kereszteződésben lévő sorban álló járművek száma. A fuzzy szabályozó kimenete pedig az aktuális zöld jelzés késleltetését adja meg, ami egy pozitív érték vagy 0 lehet.

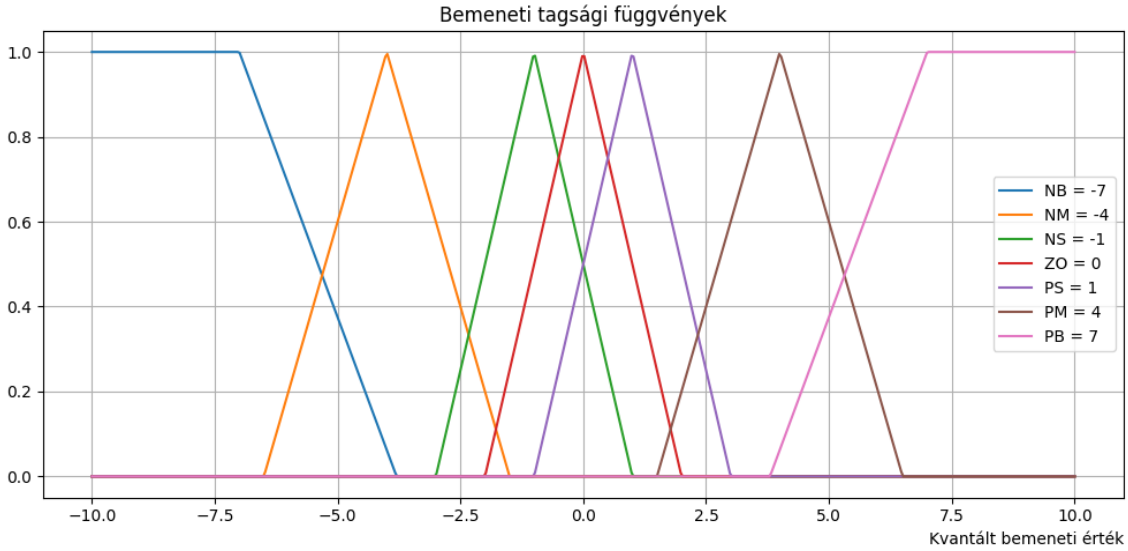


4.5.1. ábra – Szomszédos kereszteződés jelzőlámpáinak összehangolása

A két bemenetet változási tartományát [-30, 30] és [-60, 60] közé soroljuk, amit diszkretizálása az általános kanonikus formának megfelelően: {-n, -(n-1), …, -1, 0, 1, …, n-1, n}. A vezérlési pontosság követelményeinek megfelelően válasszuk ki az n értékét 7-nek, tehát a kvantálás a tartományban: {-7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7}. A diszkrét értékek és a bemeneti pontos érték közötti kapcsolat megkapható a következő képlettel:

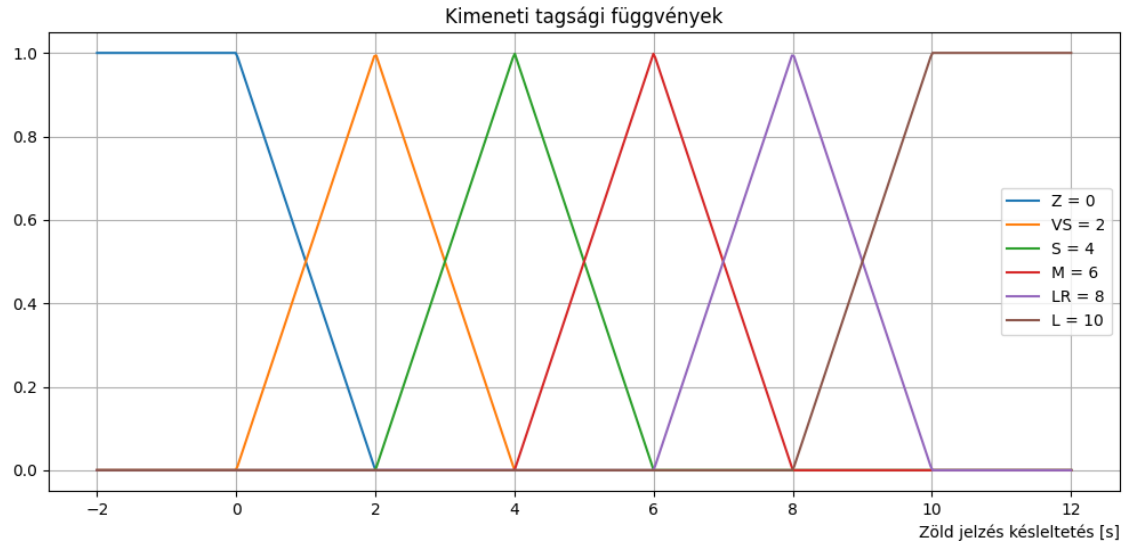
$$X = (K_x \cdot az\ aktuális\ bemeneti\ érték + 0.5) \tag{25}$$

ahol X jelöli a kvantált bemeneti értéket, K_x jelöli a kvantálási tényezőt. $K_x = \frac{7}{30}$ az első bemenet esetében és $K_x = \frac{7}{60}$ a második bemenet esetében.



4.5.2. ábra – Bemeneti tagsági függvények az összehangoló fuzzy szabályozóban

A fuzzy bemenet tagsági függvényeit a 4.5.2-es ábrán szemléltetjük, ahol mindkét bemenetnek ugyanazok a tagsági függvények felelnek meg. Látható, hogy hét tagsági függvénnyel dolgozunk és középen a 0-hoz közeli tagsági függvények jobban átfedik egymást, mint a széleken lévő függvények. A tagsági függvények középpontjai -7, -4, -1, 0, 1, 4 és 7-ben helyezkednek el.



4.5.3. ábra – Kimeneti tagsági függvények az összehangoló fuzzy szabályozóban

A kimeneti tagsági függvények ebben a szabályozásban úgy voltak meghatározva, hogy nem lehetnek negatív értékek, tehát [0, 10] -es intervallumban változhatnak. Ezt az értéket mi

később a programban még osztottuk 5-el, annak érdekében, hogy az előző fuzzy szabályozónak legyen nagyobb beleszólása a zöld jelzések módosításába. Itt hat tagsági függvényt határoztunk meg, amik egyenlő szélességgel rendelkeznek és egymástól egyenlő távolságra helyezkednek el. Ezeknek középpontjai a 0, 2, 4, 6, 8 és 10-ben vannak.

A bemeneti és kimeneti tagsági függvények rövidítéseit a cikkben megadott angol szavakból vettük át, amit a következő táblázatban foglalunk össze.

4.5.1. táblázat – Összehangoló fuzzy tagsági függvények rövidítései

Bemeneti tagsági függvények		**Kimeneti tagsági függvények**	
Negative **B**ig	**NB**	**Z**er**o**	**Z**
Negative **M**edium	**NM**	**V**ery **S**hort	**VS**
Negative **S**mall	**NS**	**S**hort	**S**
Zer**o**	**ZO**	**M**edium	**M**
Positive **S**mall	**PS**	**L**onger	**LR**
Positive **M**edian	**PM**	**L**ong	**L**
Positive **B**ig	**PB**		

A szabályok helyes meghatározása is nagyon fontos a szabályozásban, hogy az szabályok helyesen tükrözzék a szakértői tapasztalatokat és megfelelően irányítsák a rendszert. Ezért a szabálybázist is a fent említett cikknek megfelelően írtunk fel. A defuzzifikálás során az első fuzzy szabályozóhoz hasonlóan itt is a területközéppont módszerével határoztuk meg a defuzzifikált kimenetet, ami nem más, mint a zöld jelzés késleltetési ideje.

4.5.2. táblázat – Összehangoló fuzzy szabályrendszere

	NB	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
NB	Z	Z	Z	Z	Z	Z	Z
NM	Z	Z	Z	Z	Z	Z	VS
NS	Z	Z	Z	Z	Z	VS	S
ZO	Z	Z	Z	Z	VS	S	M
PS	Z	Z	Z	VS	S	M	LR
PM	Z	Z	VS	S	M	LR	L
PB	Z	VS	S	M	LR	L	L

5. Üzembe helyezés és kísérleti eredmények

5.1. Üzembe helyezési lépések

A 4.1.1-es ábrán láthattuk hogyan lehet OSM térkép segítségével egy adott helységet kiválasztani és egy alap szimulációt elindítani a kiválasztott és alapbeállításokkal. Mi a szimulációt Marosvásárhely területén végeztük, ami a következő ábrán van szemléltetve:

5.1.1. ábra – Marosvásárhely térképe a szimulációban

Ez már a szimulációba generált térkép, ahol nagyjából az egész város látható. A szimuláció indítását Python szkriptből láthattuk a 4.2.1-es ábrán. Ezt fájlt futtatva “Powershell” vagy “Command Prompt”-ból “Windows” operációs rendszerek esetén vagy “Terminal”-ból “Linux” operációs rendszereknél ugyanazt érjük el, mintha a “*.sumocfg” kiterjesztésű fájlt futtatnánk. Ha a Python fájlunk a “sumorun.py” nevet viseli, akkor a következő paranccsal tudjuk futtatni a fájlt parancssorból: python sumorun.py.

Kezdetben három fő kereszteződést irányítottunk Marosvásárhely területén: a központban lévő kereszteződést, központtól jobbra található kereszteződést, ahol a “Ștefan cel Mare” utca keresztezi a “Hosszú” utcát és a “Fortuna” buszmegálló melletti kereszteződést, amit 4.3.2-es ábrán is láthattunk. A program elején megadtuk minden kereszteződés azonosítóját egy-egy változóban, amit a szimuláció grafikus felhasználói felületéből egyszerűen ki tudunk másolni és minden kereszteződésnek külön-külön megadtuk globális változókként a következőket:

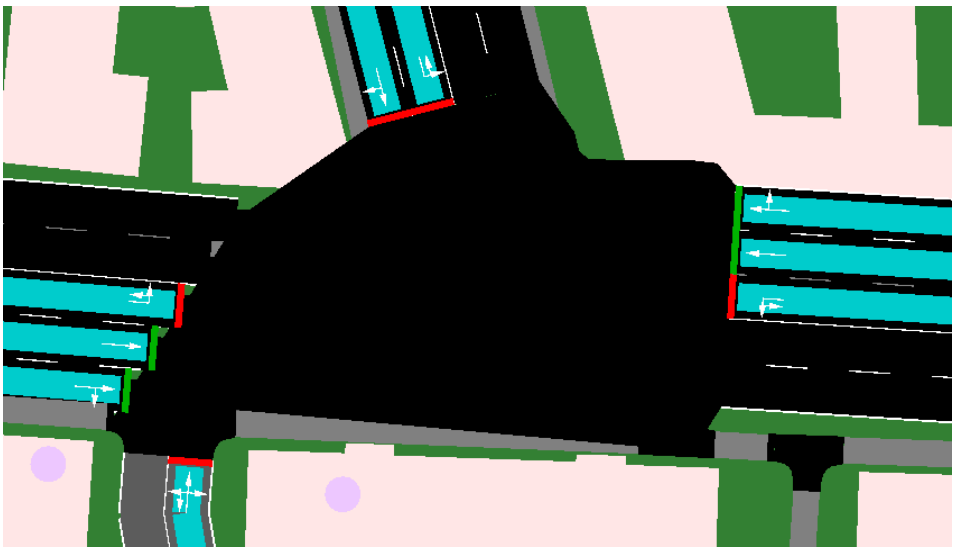
szekvenciák számát, minden szekvencia zöld idejét, minden szekvencia esetében egy $\triangle t$ értéket, amiben tároljuk az adott szekvencia zöld idejének a meghosszabbítási idejét vagy rövidítésének idejét, egy változót, amiben tároljuk azt a jövőbeli időpillanatot, amikor kell váltani a következő szekvenciára, minden szekvencia esetében a sorban álló autók számát, amivel majd később számítjuk a sorban álló autók változását és ezen kívül még minden sávterület detektor azonosítóját is eltároljuk egy-egy változóban. A while ciklusban, ahol a "step" változóval folyamatosan iterálunk minden kereszteződésben felosztott szekvencia esetében elvégezzük a következő lépéseket, amit a 5.1.2-es ábrán láthatunk.

```
if step==next_sequence and sequence==1:
                traci.trafficlight.setRedYellowGreenState(traffic_light_ref,
"gggrrrrrrrgggrrrrrr")
                delta_t1=sequence_control_ref(sequence)
                sequence_time1+=delta_t1

                if delta_t1!=0 and delta_t1%2==0:
                        sequence_time2-=delta_t1/2
                        sequence_time3-=delta_t1/2
                elif delta_t1!=0 and delta_t1%2!=0:
                        sequence_time2-=round(delta_t1/2,0)
                        delta_t1-=round(delta_t1/2,0)
                        sequence_time3-=delta_t1

                next_sequence+=sequence_time1
                sequence+=1
```

5.1.2. ábra – Jelzőlámpák irányítása

Ha a megfelelő szekvenciában vagyunk és a "step" változó elérte azt az időpillanatot, amikor az adott szekvencia ideje lejárt akkor a "traci" könyvtárban lévő "setRedYellowGreenState()" függvény segítségével módosítjuk a jelzőlámpákat a következő szekvenciának megfelelően. Ez a függvény úgy működik, hogy első paraméterként a jelzőlámpa azonosítóját kell megadni. Azt szükséges tudni a szimulációról, hogy egy kereszteződésben egy jelzőlámpaként van tekintve az összes jelzőlámpa, így egy azonosító(id) alatt szerepel az összes jelzőlámpa. Második paraméternek egy újabb string típust vár, ahol megadjuk a kívánt jelzőlámpa színeket, amilyenre szeretnénk változtatni az adott szekvenciában. A jelzőlámpa színeket az angol színek kezdőbetűivel adjuk meg: "r", "y", "g" vagy akár nagy betűkkel is meg lehet adni, amik a "piros", "sárga" és "zöld" jelzésekre utalnak. Minden egyes karakter egy-egy irányt jelöl, tehát annyi karaktert kell megadni amennyi irányba lehet haladni minden sávról

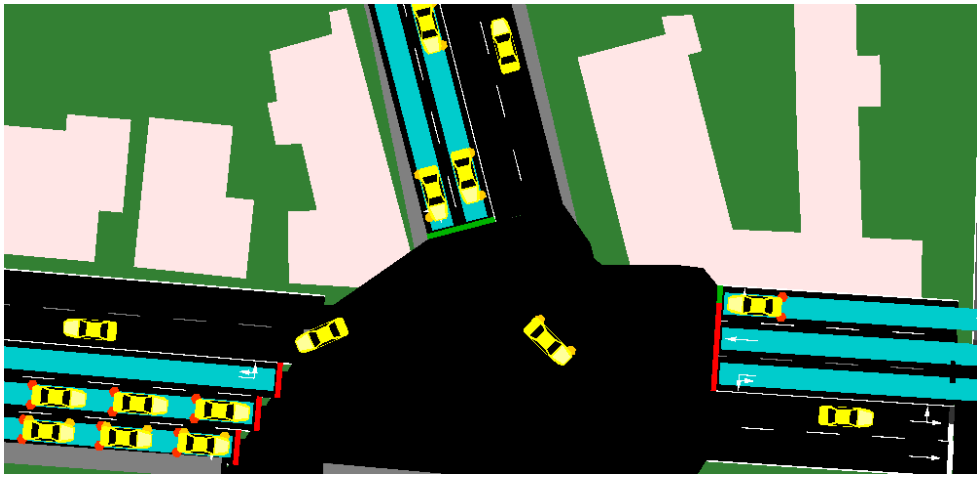
összegezve. A fenti kódrészletben megadott beállítást a 5.1.3-as ábrán láthatjuk, ahol a jobb oldalról balra haladva vannak megadva jelzőlámpák színei. A jelzőlámpák előtt lévő fehér nyilak jelzik az irányt és ezek száma meg kell egyezzen a második string paraméter hosszával.



5.1.3. ábra – Kereszteződésben lévő irányok

Ezek után meghatározzuk az aktuális szekvenciára vonatkozó $\triangle t$ értéket egy általunk megírt függvény segítségével. Ebben a függvényben lekérdezzük a sávterület detektorokkal ellátott sávokon lévő járművek számát, majd ezekből és az előző szekvenciában lekért adatokból számolunk egy változást számolunk. Ezek után meghívjuk a fuzzy irányítást, aminek paraméterként megadjuk az adott szekvenciában sorban álló autók hosszát és változását. Utána frissítjük szekvenciánként az előző lépésben lekérdezett járművek számát és visszatérítjük a fuzzy által meghatározott kimenetet, ami nem más mint a $\triangle t$ érték. Ezt az értéket hozzáadjuk az aktuális szekvencia időtartamához és ugyanazt az értéket megpróbáljuk egyenlően elosztva levonni a többi szekvencia időtartamából. A végén pedig meghatározzuk a következő időpillanatot, amikor kell a következő szekvenciára váltani és növeljük a szekvenciát, vagyis megadjuk, hogy melyik szekvencia következik.

A szimulációban többféle járműveket is lehet szimulálni, mint például: autókat, buszokat, teherautókat, vonatokat, hajókat. Ezeket a szcenárió generálás előtt kell beállítani, hogy mikkel szeretnénk a szimulációban dolgozni. Mi csak autók szimulálására állítottuk a paramétereket és így végeztük a szimulációt. A következő ábrán bemutatjuk az autókat egy kereszteződésben.



5.1.4. ábra – Autók szemléltetése a szimulációban

Az autókat sárga színnel, fekete ablakokkal és két piros hátsó féklámpákkal szemléltetik a szimulációban. Irányváltás esetén narancssárga színű jelzéssel jeleznek a megfelelő irányba az autók.

5.2. Felmerült problémák és megoldásaik

Egy kereszteződésben ahol négy útszakasz keresztezi egymást előfordult, hogy két útszakasz nagyon le volt terhelve és a másik két útszakaszon nagyon kevés autó közlekedik. Ilyenkor az a probléma merült fel, hogy a fuzzy irányítás folyamatosan növeli azoknak a szekvenciáknak a hosszát, ahol sok jármű közlekedik annyira, hogy többi szekvencia hosszát 0 vagy akár 0 másodperc alá is csökkentette. Ilyenkor az történt, hogy a jelzőlámpák bennragadtak az adott jelzésben és nem váltottak többet a szekvenciáknak megfelelően.

Ennek a problémának a megoldására minden esetben ellenőriztük, hogy a fuzzy szabályozó által visszatérített értéket kivonva a szekvencia hosszából ne legyen ne legyen 1-nél kisebb érték. Így minden szekvencia legalább 1 másodpercig fog tartani annak érdekében, hogy a fuzzy szabályozást minden lépésben tudjuk meghívni az adott szekvenciára és ha forgalmi viszonyok változnak akkor ezeknek megfelelően növelje vagy csökkentse a szabályozó a szekvenciák időtartamát. Viszont ebben az esetben is megtörtént, hogy 0 másodperc alá lecsökkent egyes szekvenciák hossza. Ez azért történt, mert azoknál a szekvenciáknál, ahol sok autó közlekedett nagyon megnövekedett a szekvencia hossza és elvette a többi szekvencia hosszából. Ezért szükséges egy-egy maximális időt is bevezetnünk, ami fölé nem engedjük a szekvenciák hosszát és minden esetben, amikor elveszünk a többi szekvencia idejéből akkor

ellenőrizzük le, hogy az elvett idő nagyobb-e a szekvencia hosszánál. Ha nagyobb akkor nem fogjuk a elvenni attól a szekvenciától, hanem az aktuális szekvenciából vonjuk le.

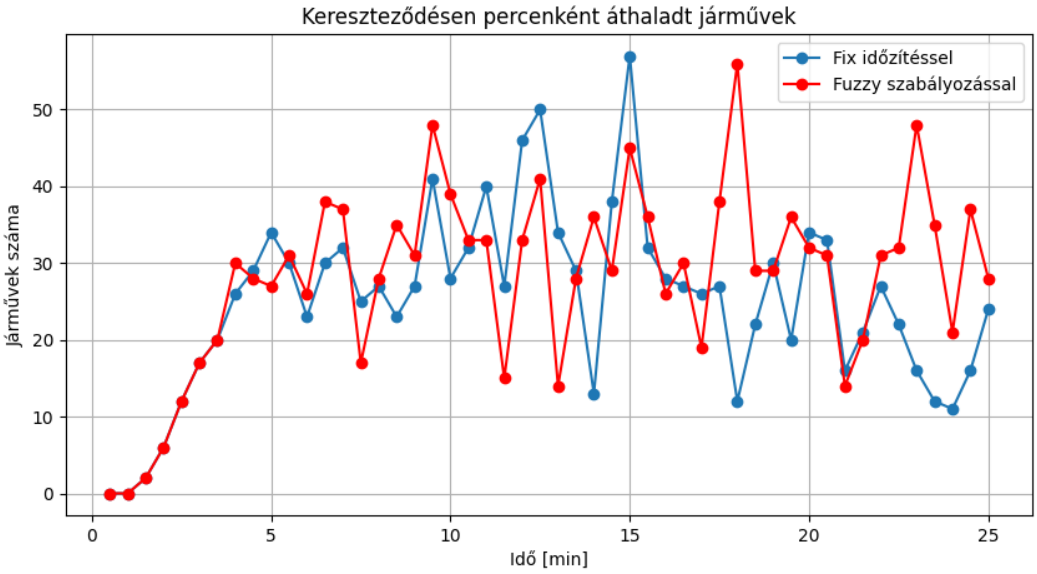
5.3. Kísérleti eredmények, mérések

A méréseket kereszteződésekre lebontva végeztük. A jelzőlámpák megfelelő irányítása azt kell eredményezze, hogy minél több autó hagyja el a kereszteződést minél rövidebb idő alatt és ne alakuljanak ki torlódások. A sávterület detektorok segítségével határoztuk meg a kereszteződésen áthaladt autók számát. Ha a sávterület detektor kevesebb autót észlel, mint amennyit az előző pillanatban, akkor azt jelenti, hogy az az autó elhagyta a jelzőlámpát, vagyis elhagyta kereszteződést. Minden másodpercben lekérdezzük a detektorokat és egy változóban tároljuk az autók számát, akik sikeresen elhagyták a kereszteződést. Minden 30. másodpercben, tehát fél percenként frissítjük a kereszteződésen áthaladt autók számát és elmentjük két változóban a 30 és a 60 másodperccel azelőtt áthaladt járművek számát. Így minden 30. másodpercben az akkori áthaladt járművek számából kivonva a 60 másodperccel előtte áthaladt járművek számát megkapjuk a percenként áthaladt járművek számát a kereszteződésben. Ezeket egy vektorba kimentjük és minden alkalommal egy “*.txt” fájlba kimentjük, amiket később grafikonon megjelenítünk.

Ezeket a méréseket elvégezzük az általunk irányított jelzőlámpákra, amiket a fuzzy szabályozással irányítottunk és ugyanezeket a méréseket elvégezzük alap beállítással, ahol a jelzőlámpák ciklikusan ismétlődő, fix időzítésű vezérlést követnek. Fontos megjegyezni azt is, hogy az alap beállításnál a jelzőlámpák nem minden esetben a szabályoknak megfelelően vannak beállítva. Ugyanabban az időben két kapnak zöld jelzést azok is akik előre vagy jobbra tartanak a kereszteződésben, de ugyanakkor azok is balra szeretnének menni. Ezért kialakulnak olyan helyzetek, hogy azok a járművek, akik balra haladnak zöld jelzést kaptak, de a kereszteződésben meg kell álljanak, mert a szemben levő útszakaszon is haladnak előre a járművek. Mi ezeket kiküszöbölve, a szabályoknak megfelelően határoztuk meg a jelzéseket és ezért a méréseket is, úgy végeztük el, hogy az egyik szimulációban az általunk megadott fix időzítést kövessenek, másik szimulációban pedig a fuzzy irányításnak megfelelően változtassák a zöld jelzések időtartamának a hosszát.

A következő ábrán láthatjuk a központban lévő kereszteződésben elvégzett mérést kirajzolva egy grafikonra a fix időzítéssel beállított szimulációban áthaladt autók számát és fuzzy irányítással áthaladt autók számát idő függvényében. Minden szekvenciának a hossza kezdetben 40 másodperc. A szimulációt 1500 másodpercig futtattuk. Az elején kisebb forgalommal indul a

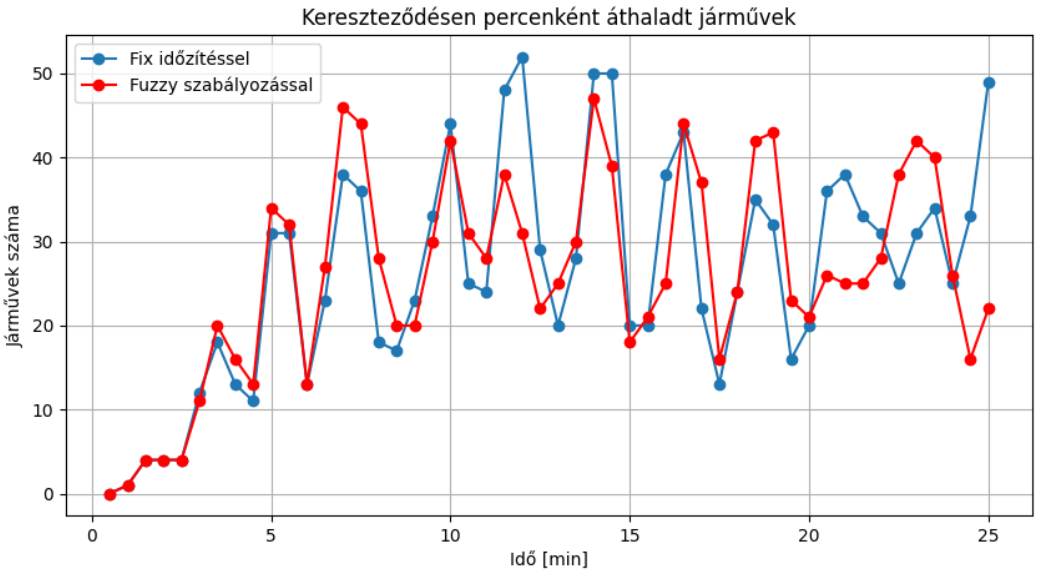
szimuláció és fokozatosan növekszik. A 200-dik másodpercben már az 1000 autót is eléri a szimuláció és később a maximális forgalom esetén 2000 autó közlekedik egész Marosvásárhely területén.



5.3.1. ábra – Központban percenként áthaladt autók

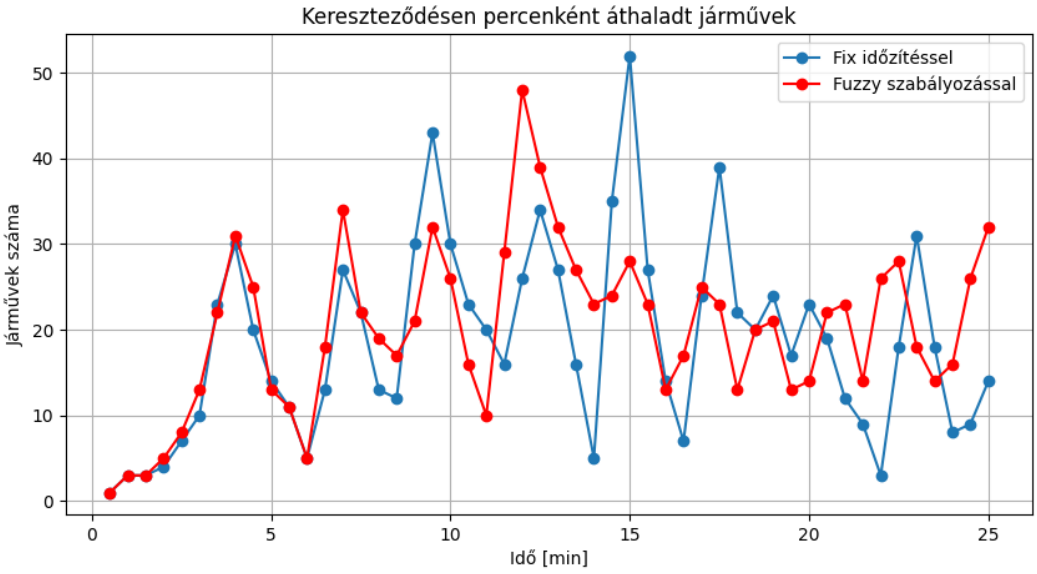
Láthatjuk, hogy mindkét szimuláció ugyanabból a kezdeti fix időzítésből indul. A fix időzítésű szimulációban megfigyelhetjük, hogy nagy ugrások jelennek meg, ami arra utal, hogy egyszer nagyon sok autó áthalad a kereszteződésen, utána pedig viszonylag kevés. A fuzzy szabályozással ezeket jobban ki tudtunk egyensúlyozni és a végére már láthatjuk, hogy sokkal több autó halad át percenként, mint fix időzítéssel.

Ezt követően a “Ștefan cel Mare” utcai kereszteződésnél végeztük a mérést. Ezt a mérést az 5.3.2-es ábra szemlélteti. Ebben az esetben három szekvenciára osztottuk a kereszteződést és ezért 45 másodpercnyi időt adtunk kezdetben minden szekvenciának. Ebben az esetben is azt figyelhetjük meg, hogy a fuzzy szabályozással irányított kereszteződésben kiegyensúlyozottabb a percenként áthaladt autók száma, tehát a fuzzy figyelembe veszi az aktuális forgalmat és annak megfelelően adaptálja a szekvenciák zöld időtartamát.



5.3.2. ábra – "Ştefan cel Mare" utcai kereszteződésben percenként áthaladt autók

Legutolsó mérésként a "Fortuna" kereszteződésnél végeztük el a szimulációs méréseket. Ebben az esetben az első hat percben nagyjából hasonló eredményeket láthatunk, viszont később a a fix időzítéses irányításban nagyon nagy ugrásokat láthatunk. Ellenkező esetben a fuzzy szabályozással, ahol nagy részben stabilan tartja az átengedett autók számát és a végén már jóval több autó halad át percenként a kereszteződésen, mint a fix időzítésnél.



5.3.3. ábra – "Fortuna" kereszteződésben percenként áthaladt autók

6. Következtetések

6.1. Megvalósítások

Összességében elmondhatjuk, hogy tanulmányoztunk több dinamikus matematikai modellt, amik differenciálegyenletek segítségével modellezik a közlekedési áramlást. Ezeket a modelleket összehasonlítottuk, elvégeztük a modellek stabilitásvizsgálatát és néhány szimulációs mérést is végeztünk a modellekre. Arra a következtetésre jutottunk, hogy a vizsgált modellek közül a FVDAM modell tükrözte a legjobban a valós közlekedési áramlást.

Ezek után a SUMO szimulációval ismerkedhettünk meg. A SUMO szimulációval szimuláltuk Marosvásárhely területét. Felépítettünk egy fuzzy szabályozót Python nyelvben, amivel a kereszteződésekben a jelzőlámpák zöld idejét befolyásoltuk az aktuális forgalomnak megfelelően. A TraCI interfész segítségével a Python nyelvben megírt fuzzy szabályozót bele tudtuk építeni a szimulációba. Ezt követően még egy fuzzy szabályozót felépítettünk, ami a szomszédos kereszteződések összehangolásában játszott szerepet. Ennek segítségével sikerült három kereszteződést összehangolni, vagyis a Fortuna kereszteződéstől egy zöldhullámot kialakítani a központ fele. A dolgozat végén pedig méréseket végeztünk, amik kimutatták, hogy a fuzzy szabályozással vezérelt kereszteződések sokkal kiegyensúlyozottabbak, képesek adaptálódni a forgalomnak megfelelően és a kereszteződésen percenként áthaladt autók száma is jóval megnövekedik a fuzzy szabályozás után, mint a fix időzítésű vezérlés esetén.

6.2. Továbbfejlesztési lehetőségek

Továbbfejlesztési lehetőségként a város összes jelzőlámpával irányított kereszteződését lehetne irányítani fuzzy szabályozással, amivel sokkal hatékonyabb irányítást lehetne megvalósítani, mivel minden jelzőlámpás kereszteződés irányítva lenne a forgalomnak megfelelően és sokkal kisebb lenne az esélye annak, hogy torlódások alakuljanak ki.

Egy másik fejlesztési lehetőségként hosszabb útvonalon vagy akár több zöldhullám effektust lehetne beépíteni az irányításba, ezzel is csökkentve annak esélyét, hogy a torlódások alakuljanak ki.

7. Irodalomjegyzék

[1] B. De Schutter, H. Hellendoorn, A. Hegyi, M. van den Berg, and S.K. Zegeye, “Modelbased control of intelligent traffic networks,” Chapter 11 in Intelligent Infrastructures (R.R. Negenborn, Z. Lukszo, and H. Hellendoorn, eds.), vol. 42 of Intelligent Systems, Control and Automation: Science and Engineering, Dordrecht, The Netherlands: Springer, ISBN 978-90- 481-3598-1, pp. 277–310, 2010.

[2] Lili Zhang , Qi Zhao , PeiYu , Jing Li , DiYao , XinzheWang , LiWang & Lingyu Zhang, “Research on integrated simulation platform for urban trafc control connecting simulation and practice”, 2022.

[3] Tettamanti T., Varga I., “MPC alapú, elosztott városi forgalomirányító rendszer”, 2009.

[4] M. Bando and K. Hasebe, A. Nakayama, A. Shibata, Y. Sugiyarna, “Dynamical model of traffic congestion and numerical simulation”, 1995.

[5] Rui Jiang, Qingsong Wu, Zuojin Zhu, “Full velocity difference model for a car-following theory”, 2001.

[6] Shaowei Yu , Qingling Liu, Xiuhai Li, “Full velocity difference and acceleration model for a car-following theory”, 2012.

[7] Dirk Helbing and Benno Tilch, “Generalized force model of traffic dynamics”, 1998.

[8] Jing Zhang, Bo Wang, Shubin Li, Tao Sun, Tao Wang, “Modeling and application analysis of car-following model with predictive headway variation”, 2019.

[9] Tan Kok Khiang, Marzuki Khalid, Rubiyah Yusof, “Intelligent Traffic Lights Control By Fuzzy Logic”, 1997.

[10] Javed Alam, Dr. M. K. Pandey, Husain Ahmed, “Intellegent Traffic Light Control System for Isolated Intersection Using Fuzzy Logic”, 2013.

[11] Yi Zhang, Gengsheng Huang, “Based on road green wave effect of collaborative strategy of signal timing fuzzy control”, 2013.

8. Függelék

```xml
<?xml version="1.0" encoding="UTF-8"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">

    <input>
        <net-file value="osm.net.xml.gz"/>
        <route-files value="osm.passenger.trips.xml"/>
        <additional-files value="osm.poly.xml.gz"/>
    </input>

    <processing>
        <ignore-route-errors value="true"/>
    </processing>

    <routing>
        <device.rerouting.adaptation-steps value="18"/>
        <device.rerouting.adaptation-interval value="10"/>
    </routing>

    <report>
        <verbose value="true"/>
        <duration-log.statistics value="true"/>
        <no-step-log value="true"/>
    </report>

    <gui_only>
        <gui-settings-file value="osm.view.xml"/>
    </gui_only>

</configuration>
```

F. 1. ábra – SUMO konfigurációs fájl

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan

Conf. dr. ing. Domokos József

Vizat director departament

Ș.l. dr. ing Szabó László Zsolt