**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: Absolvent:**

**Prof.dr.ing. Dávid László, Mikló József-Péter**

**Prof.dr.ing. Farkas Csaba**

**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:  **ș.l. dr. ing. Dávid László** | | Candidat: **Mikló József-Péter**  Anul absolvirii: **2024** |
| **a) Tema lucrării de licență:**  CONTROLUL TRAFICULUI URBAN IMPLEMENTAT CU SISTEM DE REGLARE FUZZY  **b) Problemele principale tratate:**  **-** Studiu bibliografic 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., Reţele 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 responsabilului  programului de studiu | Semnătura coordonatorului  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

A dolgozat 1 oldalas kivonata román nyelven (Times New Roman betűtípus 1,5 sorköz) .

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM**

**MAROSVÁSÁRHELYI KAR**

**AUTOMATIKA ÉS ALKALMAZOTT INFORMATIKA SZAK**

**FUZZY SZABÁLYOZÁSSAL MEGVALÓSÍTOTT VÁROSI FORGALOMIRÁNYÍTÓ RENDSZER**

**DIPLOMADOLGOZAT**

**Témavezető: Végzős hallgató:**

**Dr. Dávid László, egyetemi tanár Mikló József-Péter**

**Dr. Farkas Csaba, egyetemi tanár**

**2024**

Kivonat

Napjainkban nagyon sok jármű közlekedik városokban, illetve autópályákon. Amikor a járművek száma meghaladja az utak kapacitását forgalmi torlódások jönnek létre. 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. Többféle megközelítések is születtek az évek során a forgalomirányításra, többek között: mesterséges intelligenciával, Fuzzy szabályozással, modell prediktív irányítással.

A dolgozatban különböző dinamikus matematikai modelleket tanulmányoztunk, amelyek stabilitása meghatározza a torlódások kialakulását és 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.

***Kulcsszavak***: forgalomirányítás, Fuzzy, SUMO, Python

Abstract

A dolgozat angol kivonata 150-200 szó között (Times New Roman betűtípus 1,5 sorköz).

***Keywords***: motion detection, motion tracking, biometry

**Tartalomjegyzék**

[1. Bevezető 11](#_Toc169474037)

[2. Elméleti megalapozás és szakirodalmi tanulmány 12](#_Toc169474038)

[2.1. Közlekedési áramlás matematikai modellezése 12](#_Toc169474039)

[2.2. A dinamikus modellek stabilitásvizsgálata 15](#_Toc169474040)

[2.3. FVDAM és FVDM modellek szimulálása 19](#_Toc169474041)

[3. Célkitűzések 23](#_Toc169474042)

[4. Irányítás szimulációban 24](#_Toc169474043)

[4.1. Szimulációs szoftver 24](#_Toc169474044)

[4.2. TraCI 26](#_Toc169474045)

[4.3. Fuzzy irányítás 27](#_Toc169474046)

[5. Üzembe helyezés és kísérleti eredmények 38](#_Toc169474047)

[5.1. Üzembe helyezési lépések 38](#_Toc169474048)

[5.2. Felmerült problémák és megoldásaik 40](#_Toc169474049)

[5.3. Kísérleti eredmények, mérések 40](#_Toc169474050)

[6. Következtetések 41](#_Toc169474051)

[6.1. Megvalósítások 41](#_Toc169474052)

[6.2. Hasonló rendszerekkel való összehasonlítás 41](#_Toc169474053)

[6.3. Továbbfejlesztési lehetőségek 41](#_Toc169474054)

[7. Irodalomjegyzék 42](#_Toc169474055)

[8. Függelék 42](#_Toc169474056)

**Ábrák jegyzéke**

[2.1.1. ábra – Sebességfüggvény ábrázolása 13](#_Toc169559968)

[2.1.2. ábra – Optimális sebességfüggvény ábrázolása 14](#_Toc169559969)

[2.3.1. ábra - Autók pozíciója (FVDM) 20](#_Toc169559970)

[2.3.2. ábra - Autók pozíciója (FVDAM) 20](#_Toc169559971)

[2.3.3. ábra - Autók sebessége (FVDM) 21](#_Toc169559972)

[2.3.4. ábra - Autók sebessége (FVDAM) 21](#_Toc169559973)

[2.3.5. ábra - Autók gyorsulása (FVDM) 22](#_Toc169559974)

[2.3.6. ábra - Autók gyorsulása (FVDAM) 22](#_Toc169559975)

[4.1.1. ábra – Szcenárió generálás OSM térképen 25](#_Toc169559976)

[4.1.2. ábra – SUMO konfigurációs fájl 26](#_Toc169559977)

[4.2.1. ábra – SUMO szimuláció indítása Python környezetben 27](#_Toc169559978)

[4.2.2. ábra – Sávterület detektor létrehozása 27](#_Toc169559979)

[*4.3.1. ábra – Fuzzy irányítás lépései* 29](#_Toc169559980)

[4.3.2. ábra – Marosvásárhelyi kereszteződés felosztása szekvenciákra 30](#_Toc169559981)

[4.3.3. ábra – Sávterület detektorok a szimulációban 31](#_Toc169559982)

[4.3.4. ábra – Fuzzy irányítás osztálydiagramja 32](#_Toc169559983)

[4.3.5. ábra – Bemeneti tagsági függvények ábrázolása 33](#_Toc169559984)

[4.3.6. ábra – Kimeneti tagsági függvények ábrázolása 34](#_Toc169559985)

[4.3.7. ábra – Bemeneti tagsági függvények levágása 36](#_Toc169559986)

[4.3.8. ábra – Levágott kimeneti tagsági függvények 37](#_Toc169559987)

[5.1.1. ábra – Marosvásárhely térképe a szimulációban 38](#_Toc169559988)

[5.1.2. ábra – Jelzőlámpák irányítása 39](#_Toc169559989)

[5.1.3. ábra – Kereszteződésben lévő irányok 40](#_Toc169559990)

[5.1.4. ábra – Autók szemléltetése a szimulációban 41](#_Toc169559991)

[5.3.1. ábra – Központban áthaladt autók 43](#_Toc169559992)

[5.3.2. ábra – Református kollégium utcai kereszteződésben áthaladt autók 43](#_Toc169559993)

[5.3.3. ábra – Fortuna kereszteződésben áthaladt autók 44](#_Toc169559994)

**Táblázatok jegyzéke**

[4.3.1. táblázat: Fuzzy tagsági függvények rövidítései 34](#_Toc169869824)

[4.3.2. táblázat: Fuzzy szabályok 36](#_Toc169869825)

# 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]](#footnote-1), SCATS[[2]](#footnote-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]](#footnote-3) [3].

A dolgozat két fő részből tevődik össze:

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

# Elméleti megalapozás és szakirodalmi tanulmány

## 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. Bandoés 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:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

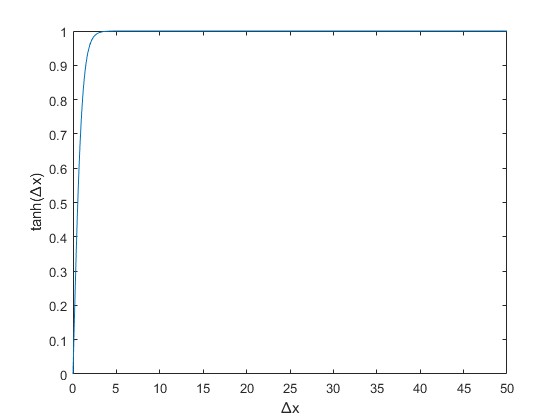
ahol jelöli az n-dik autó gyorsulását, -val jelöltük a vezető érzékenységét , , jelöli az (n+1)-dik és az n-dik jármű közötti távolságot, jelöli a sebességfüggvényt és pedig az n-dik jármű sebességét.

A sebességfüggvény a következő egszerű képlettel adhatjuk meg:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

ahol a járművek közötti távolságot jelöli.

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



2.1.1. ábra – Sebességfüggvény ábrázolása

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:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

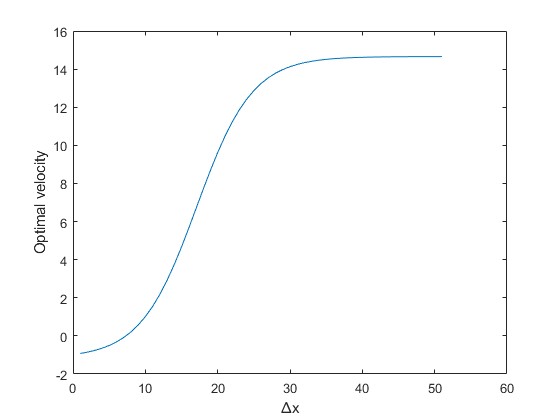
ahol paraméterek megfelelnek az (1) képletben lévő paramétereknek, és még kiegészülnek a következőkkel: jelöli az n-dik jármű sebességét, jelöli az (n+1)-dik és az n-dik jármű sebességkülönbségét, a vezető érzékenységi együtthatója a sebességkülönbséghez , V(.) az optimális sebességfüggvény.

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

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

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

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



2.1.2. ábra – Optimális sebességfüggvény ábrázolása

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.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

ahol a jelölések megegyeznek az (1) és (2) egyenletekben használt paraméterekkel, kiegészítve -vel, ami jelöli az (n+1)-dik jármű (vezető) gyorsulását, k pedig a követő autó érzékenységi együtthatóját jelöli, . 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:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Az (6) egyenletet Taylor sorbafejtéssel leegyszerűsíthetjük a következőképpen: , amiből kapjuk a (7) egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

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

## 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 egy nemlineáris dinamikus modell és =0 minden –ra. Ha létezik egy olyan Lyapunov függvény, amely deriválható egyensúlyi pont körül és az alábbi feltételeket teljesíti, akkor beszélhetünk stabil rendszerről:

1. pozitív definit: és
2. negatív szemidefinit:
3. negatív definit:

Ha az 1. és a 2. feltétel teljesül, akkor egyensúlyi pontja a rendszernek stabil és ha az 1. és 3. feltétel teljesül akkor 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: , =0 minden –ra. Legyen az Jacobi mátrixa, melyet x=0 kezdőpontban értékelünk ki. Minden t-re kapunk egy maradékot, amit a következőképpen írhatunk le: . Mivel a maradék nem biztos, hogy egyenletesen közelít a 0-hoz, ezért egy erősebb feltételre van szükség: . Ha ez az egyenlet teljesül, akkor 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 stabilitásávizsgálatával 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:

|  |  |  |
| --- | --- | --- |
|  | , | (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 egy kis zavar amit hozzáadunk a stabil egyenlethez.

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

A (8) és (9) egyenletet behelyettesítve az (1) egyenletbe, majd leegyszerűsítve és megkapjuk a következő összefüggést:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

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

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

Fourier transzformáció után a következővel helyettesítve: , a deriváltak pedig: , , amit behelyettesítve a (11) egyenletbe majd az egész egyenletet elosztva -el a következő összefüggést kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

ahol egy komplex számot jelöl, .

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:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Ha teljesül a (13)-as feltétel, akkor a felírt dinamikus modell stabil, ha akkor a modell a stabilitás határán van és ha 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: , majd Taylor sorbafejtés után a (11) egyenlet helyett a következőt kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Előzőhöz képest még bejött tag az egyenletbe. A (12) egyenlet pedig a következőképpen változik meg:

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

Az előbbi modellhez képest itt még bejött egy tag az egyenletbe. Behelyettesítve és a (15) egyenletbe kapjuk a következő egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

Innen pedig megkapjuk a stabilitási feltételt:

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

Itt a stabilitáshoz már hozzájárul a 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: , Taylor sorbafejtés után pedig:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

Látható, hogy itt már bejött két plusz tag az első modellhez képest és egy plusz tag a FVDM modellhez képest (). behelyettesítve a modellbe a következőképpen alakul az egyenlet:

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

A FVDM modellhez viszonyítva a meg lett szorozva még -val.

Behelyettesítve és a (19) egyenletbe kapjuk a következő egyenletet:

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

Innen pedig megkapjuk a semleges stabilitási feltételt:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

Itt már teljesen megváltozik a stabilitási feltétel, de k=0-ra a FVDAM leegyszerüsödik 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: , majd Taylor sorba fejtés után a következőképpen alakul az egyenlet:

|  |  |  |
| --- | --- | --- |
|  |  | (22) |

Jing Zhang és munkatársai cikkét tanulmányozva egy hibára bukkantunk. A (22) egyenlet náluk helytelenül szerepel: helyett szerepel, amivel a későbbi eredményeket sem tudjuk megkapni.

behelyettesítve a (22) egyenletbe következőt összefüggést kapjuk:

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

FVDM modelnél a (15) egyenlettel összehasonlítva ebben az egyenletben az még meg van szorozva -vel.

és behelyettesítve a (23) egyenletbe megkapjuk a stabilitási feltételt, ami ebben az esetben a következő:

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

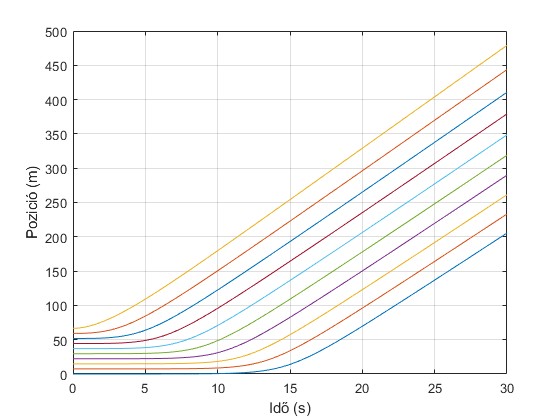
## FVDAM és FVDM modellek szimulálása

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 –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:, , , , , , 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 modell esetében, vagyis amikor k=0 helyettesítünk az (5)-ös egyenlet FVDAM modellbe, akkor megkapkapjuk a (3)-as egyenletet, ami a FVDM modell.

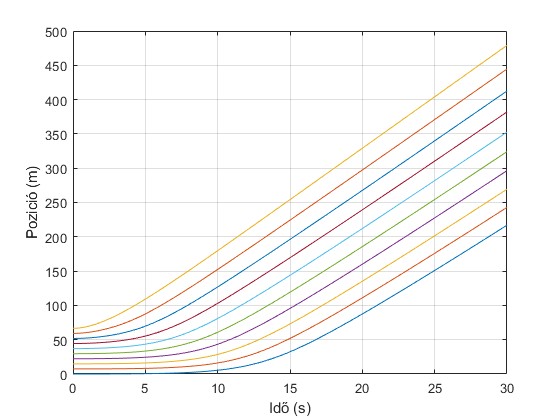
Távolság egyre növekszik az autók között, ahogy telik az idő!!!



2.3.1. ábra - Autók pozíciója (FVDM)

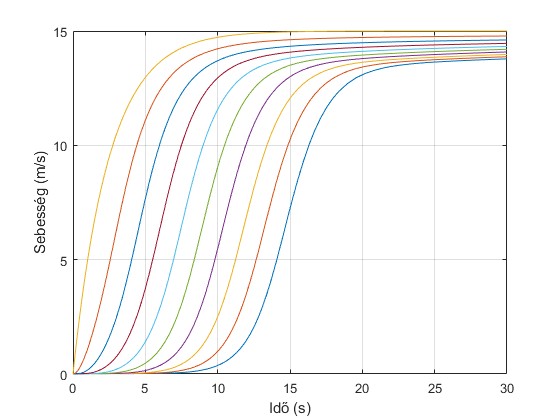
+tavolsagok megjelenitese

A 2.3.2 –es ábra is az autók pozícióját ábrázolja idő függvényében a FVDAM modell esetében. A FVDAM modellek esetében k=0.5-re választottuk a cikknek megfelelően. Az ábrán észrevehető, hogy az autók már időben hamarabb változtatják a pozíciójukat, tehát hamarabb reagálnak az előttük haladó autóra.



2.3.2. ábra - Autók pozíciója (FVDAM)

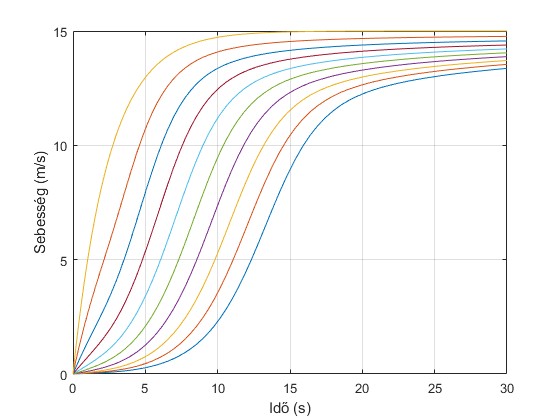
A következő szimulációs ábrán az autók sebessége van ábrázolva idő függvényében a FVDM modell esetében. Minden járműnek be kellene állnia egy konstans sebességre.



.. ábra - Autók sebessége (FVDM)

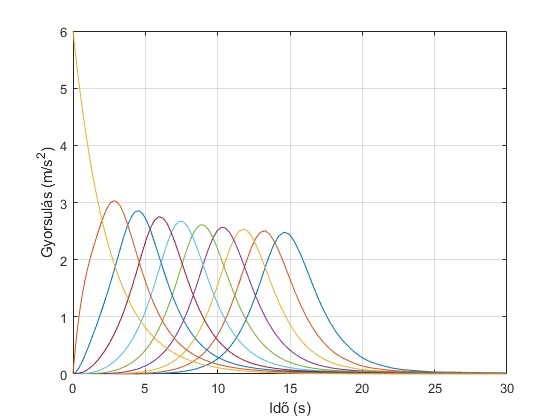
Sebesseg egyseges kellene legyen!!! Ezert lesz nagyobb a tavolsag kulonbseg

A 2.3.4-es ábrán úgyszintén az autók sebességét ábrázoltuk idő függvényében a FVDAM modellre. Mivel a 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. Ezzel azt eredményezzük, hogy a követő autók nem fognak hirtelen nagy sebességre kapcsolni és nem fognak hirtelen nagyot fékezni sem.



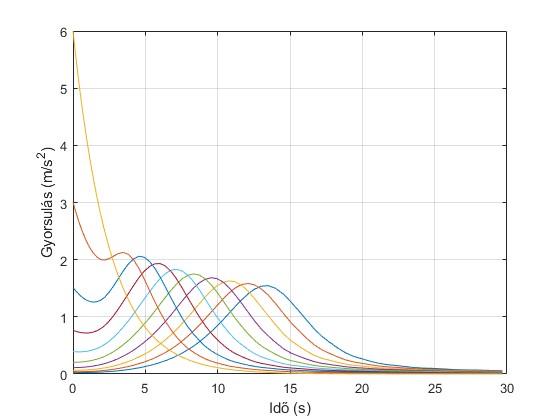
2.3.4. ábra - Autók sebessége (FVDAM)

A 2.3.5-ös ábrán már az autók gyorsulását láthatjuk idő függvényében ábrázolva a FVDM modellre. A vezető autó gyorsulása -ből indul és exponenciálisan csökken 0-ba, míg a többi autó gyorsulás -ből exponenciálisan növekszik egy ideig és utána exponenciálisan csökken 0-ba, ami minden követő autó esetén egy-egy Gauss görbét határoz meg.



2.3.5. ábra - Autók gyorsulása (FVDM)

A FVDAM modell esetében ugyanúgy a gyorsulást reprezentáljuk idő függvényében a 2.3.6-os ábrán.



.. ábra - Autók gyorsulása (FVDAM)

Itt már látható, hogy az elején lévő követő autók már egy adott gyorsulással indulnak t=0 időpillanatban és sokkal jobban követik a vezető autót. A hátrább lévő autók már közel, de nem teljesen 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.

# 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:

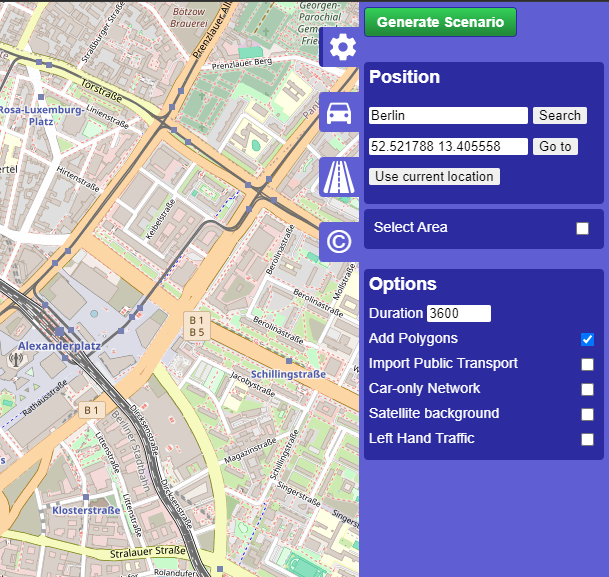
* torlódások elkerülése és enyhítése
* megfelelő szabályozás nagy és kis forgalom esetén is
* zöldhullám kialakítása
* elkerülő utak ütemezése

# Irányítás szimulációban

## Szimulációs szoftver

Közlekedési hálózatok modellezésére és elemzésére a SUMO[[4]](#footnote-4), MATSim[[5]](#footnote-5), VISSIM[[6]](#footnote-6) vagy AIMSUN[[7]](#footnote-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]](#footnote-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]](#footnote-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.



4.1.1. 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 következőképpen néz ki:

<?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>

4.1.2. ábra – SUMO konfigurációs fájl

A rendezettebb munka érdekében az xml fájlokat egy mappába helyeztük, majd az elérési útvonalakat ennek megfelelően megváltoztattuk a konfigurációs fájlban is.

## 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.

<?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

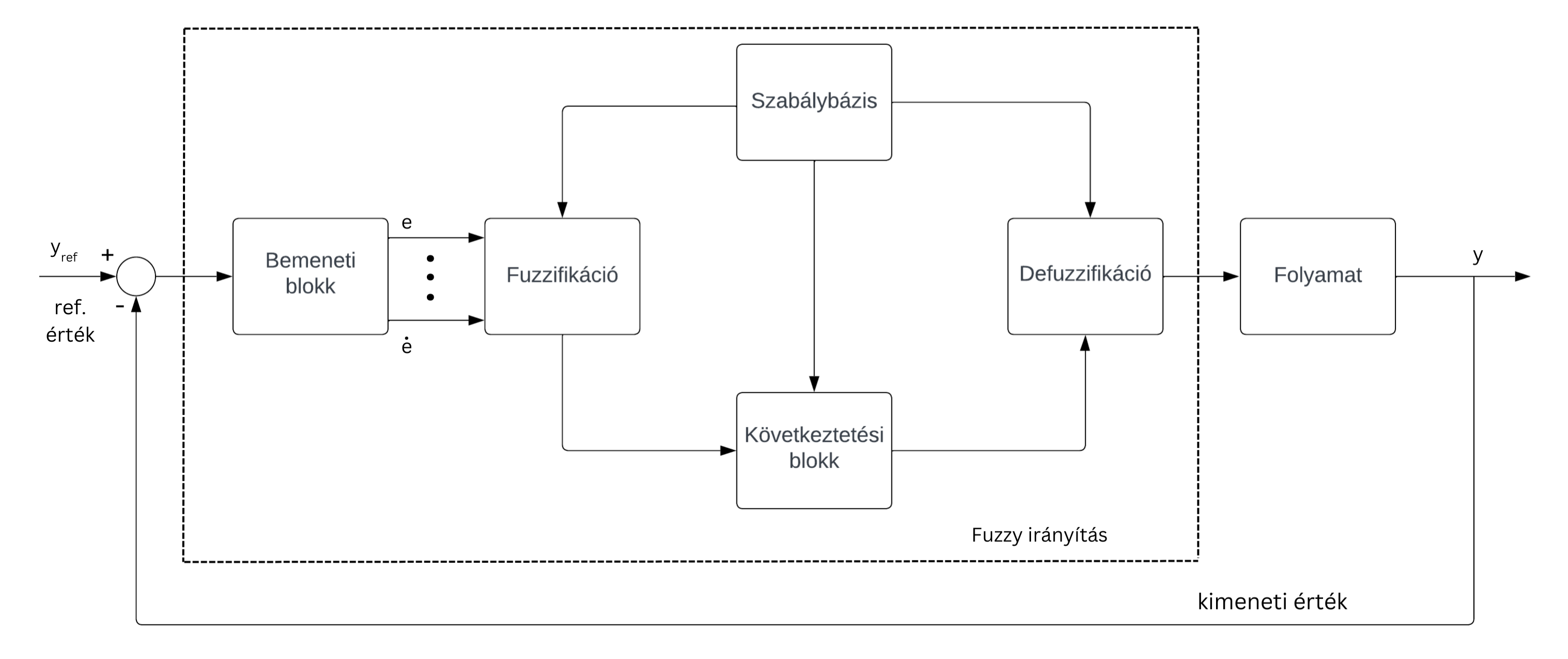
A 4.2.2-es ábrán egy detektor létrehozását láthajuk, ami egy megadott hosszúságú sávon figyeli a járművek számát. A detektor létrehozásához szükséges megadnunk egy azonosítót (id-t), a megfigyelt sávok azonosítójait(egymás után lévő sávokat össze lehet kötni egy detektorba), a sáv kiinduló és végpontját, ameddig szeretnénk, hogy tartson a detektálás egy kimeneti fájlt és a jelzőlámpa azonosítóját.

## 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. [9]

Javed Alam és társai cikkében [9], valamint Tan Kok Khiang 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 4.3.1-es ábrán láthatjuk.



*4.3.1. 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.

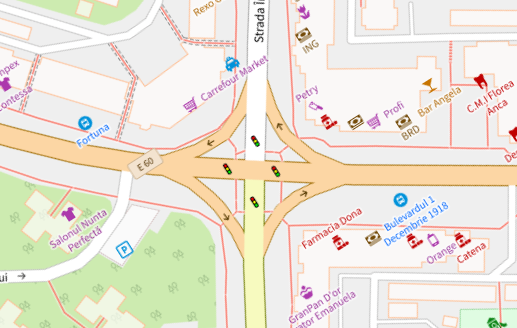
A szabálybázisban megadjuk a szabályokat ami alapján szeretnénk, hogy döntsön a rendszer. Ezek a szabályok emberi gondolkodásmódon alapulnak. Például ha jelzőlámpáknál sorban álló autók hossza az egyik bemenet: “hosszú” és a másik bemenet a változása: “növekszik”, akkor a fuzzifikált kimenetünk a zöld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

A következtetés során a szabályrendszer segítéségével megkapjuk a fuzzifikált kimenetet és annak a mértékét. A fuzzifikált kimenetek ugyanugy tagsági függvények, de egyszerűbb megvalósításban singletonok is lehetnek.

A defuzzifikáció az a lépés amikor fuzzifikált kimeneteket defuzzifikáljuk és megkapjuk a pontos 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 irányítással. Ezt úgy valósítottuk meg, hogy felosztottuk a kereszteződést három vagy négy szekvenciára a kereszteződés bonyolultságának megfelelően. Ha a 4.3.2-es ábrát tekintjük, ami a “Fortuna” buszmegálló melletti kereszteződés Marosvásárhelyen, akkor ezt a kereszteződést négy szekvenciára bonthatjuk fel:

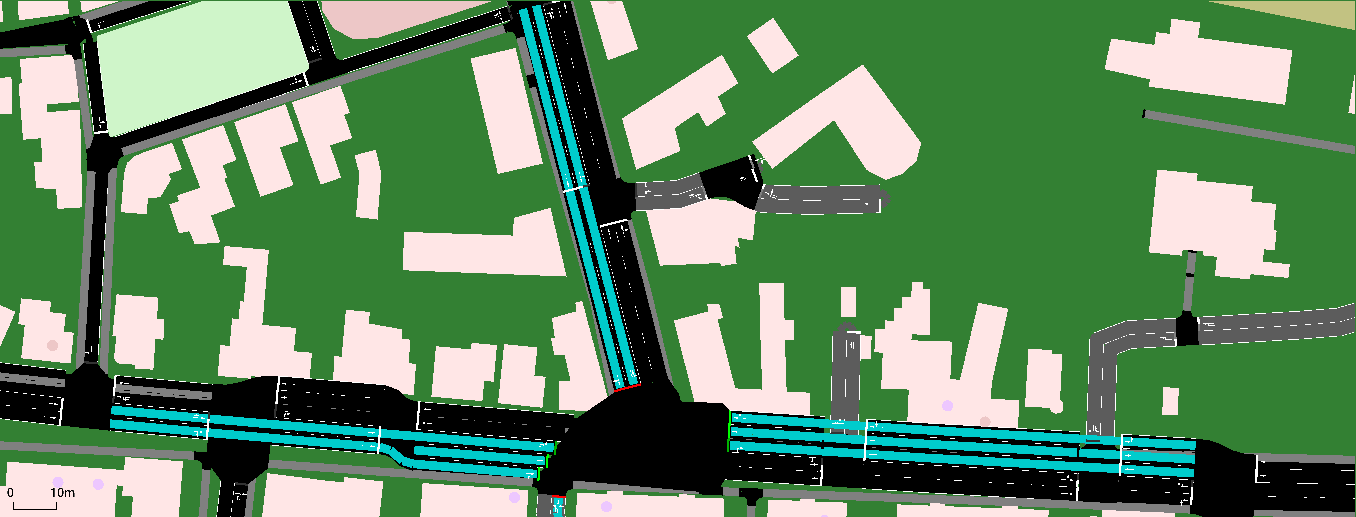
1. nyugatról és keletről közlekedő járművek kapnak előre és jobbra zöld jelzést
2. nyugatról és északról haladó járművek kapnak balra zöld jelzést és az északi és déli oldalról érkező járművek is kapnak jobbra zöld jelzést
3. északról és délről közlekedő járművek kapnak előre és jobbra zöld jelzést
4.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



4.3.2. ábra – Marosvásárhelyi kereszteződés felosztása szekvenciákra

Ezek után minden szekvenciának kezdetben megadtunk egy időtartamot, úgy hogy az elején minden szekvencia időtartama egyenlő legyen és ezt az irányítással csökkentjük vagy növeljük a forgalomnak megfelelően.

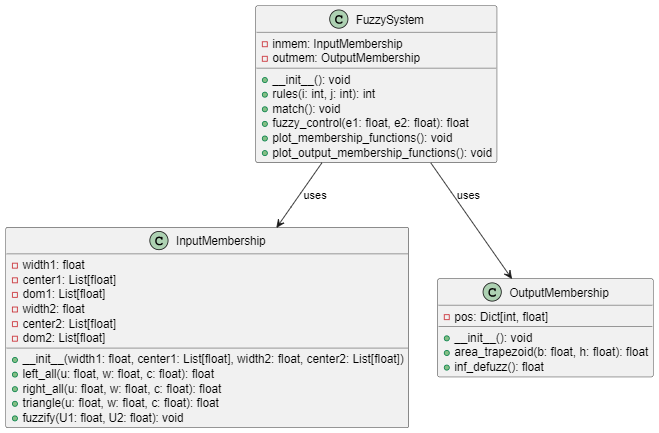
A teljes Fuzzy irányítást mi építettük fel egy Python scriptben. Úgy valósítottuk meg, hogy két bemeneti értéket kell megadjuk: az aktuális szekvenciában lévő várakozó járművek számát és ezeknek a változását. Kimenetként pedig visszatérít egy értéket, ami megmondja, hogy hány másodperccel növeljük vagy 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.



4.3.3. 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 4.3.3-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.

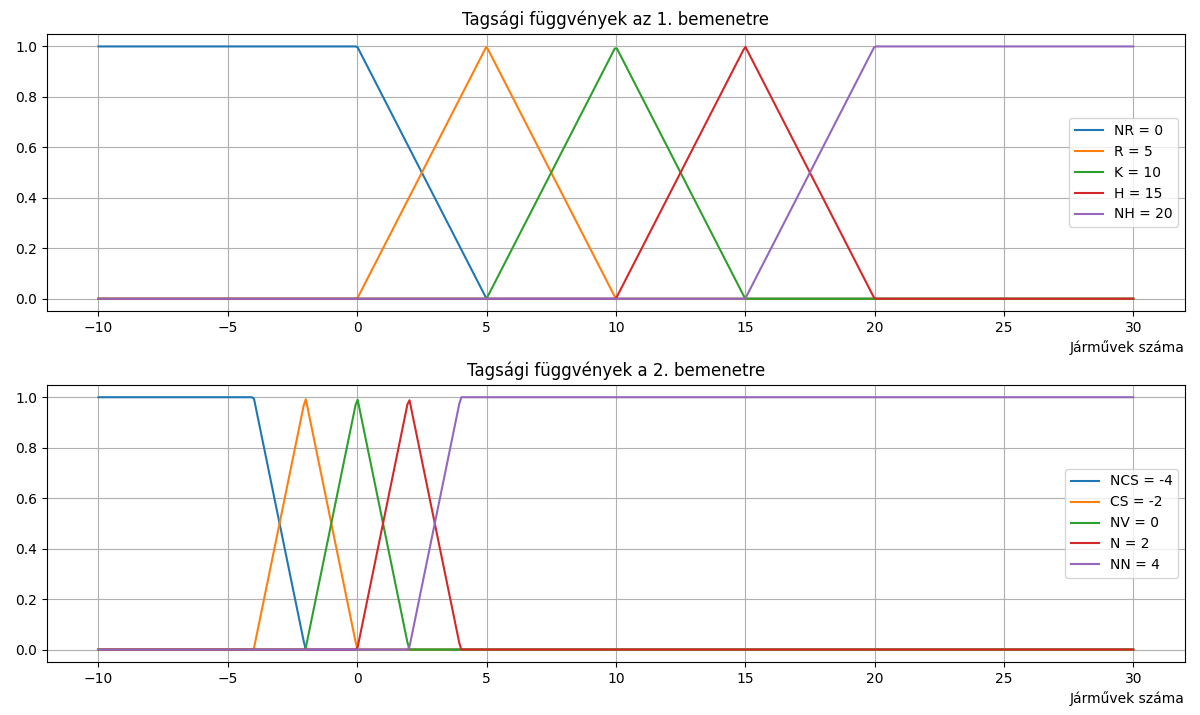


4.3.4. ábra – Fuzzy irányítás osztálydiagramja

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 4.3.5-ös ábrán tüntettük fel.



4.3.5. ábra – Bemeneti tagsági függvények ábrázolása

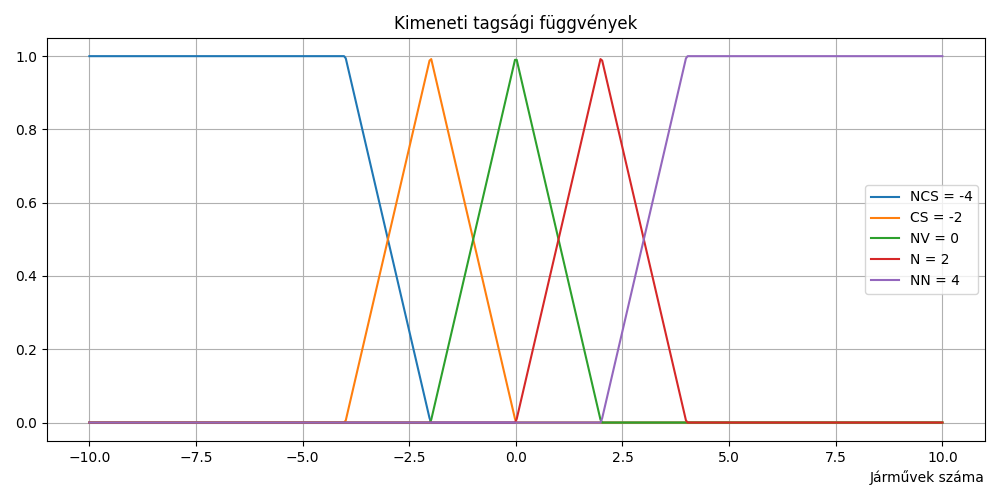
Az első bemenet tagsági függvényei vagy másképp mondva bemeneti változói a szekvencia hosszára utalnak. A tagsági függvények középpontjai pedig a **0**, **5**, **10**, **15** és **20** -ban helyezkednek el. A második bemeneti tagsági függvények a szekvencia hosszának a változását jelöli és középpontjai pedig a **-4**, **-2**, **0**, **2** és **4** -ben helyezkednek el. A bemeneti és a kimeneti tagsági függvények rövidítéseit a 4.3.1-es táblázatban szemléltetjük.

4.3.1. táblázat: Fuzzy tagsági függvények rövidítései

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Első bemeneti tagsági függvények** | | **Második bemeneti tagsági függvények** | | **Kimeneti tagsági függvények** | |
| **N**agyon **R**övid | **NR** | **NCS** | **N**agyon **Cs**ökken | **NCS** | **N**agyon **Cs**ökkentjük |
| **R**övid | **R** | **CS** | **Cs**ökken | **CS** | **Cs**ökkentjük |
| **K**özepes | **K** | **NV** | **N**em **V**áltozik | **NV** | **N**em **V**áltoztatjuk |
| **H**osszú | **H** | **N** | **N**övekszik | **N** | **N**öveljük |
| **N**agyon **H**osszú | **NH** | **NN** | **N**agyon **N**övekszik | **NN** | **N**agyon **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” azonósítójú 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 4.3.6-os ábrán láthatjuk a kimeneti tagsági függvények ábrázolását.



4.3.6. 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 ugyszintén 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. A szabályokat az 1. 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:

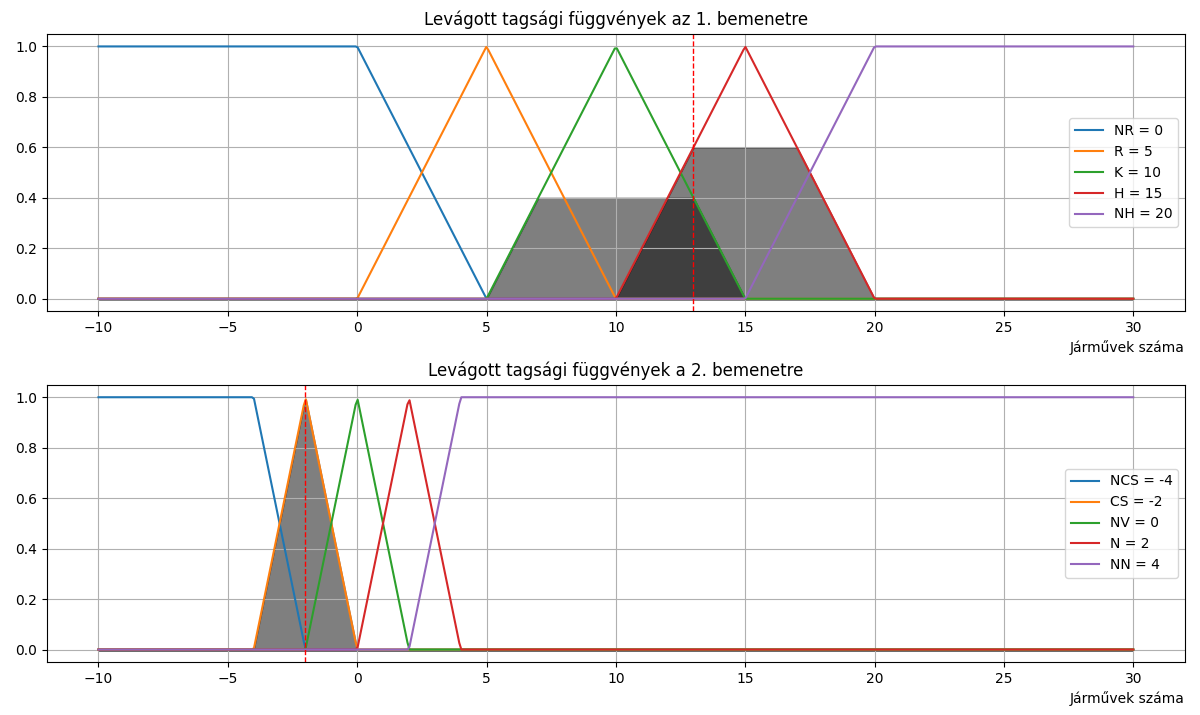
* 2. sor 2. oszlop: Ha a szekvencia hossza nagyon rövid (NR) és a változása nagyon csökken (NCS) akkor a nagyon csökkentjük (NCS) a zöld időt
* 5. sor 4. oszlop: Ha a szekvencia hossza közepes (K) és a változása növekszik (N) akkor növeljük (N) a zöld jelzést.

.. 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 |

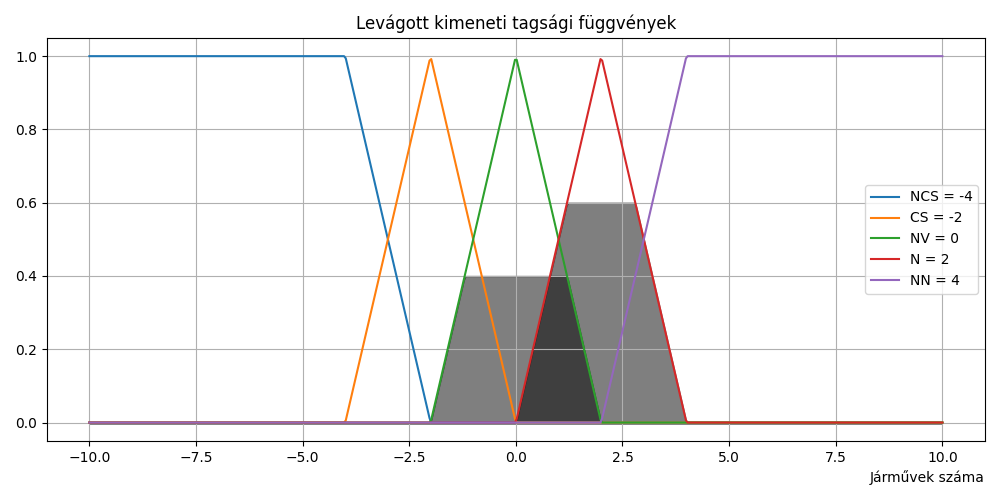
A fuzzy szabályozás következtetési mechanizmusa hasonlít az emberi gondolkodási folyamatra. Ez az a pont, ahol fuzzy szabályozás technológiája a mesterséges intelligenciával társul. Az emberek tudattalanul használnak szabályokat a cselekedeteik végrehajtásában. Például egy közlekedést irányító rendőr, 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. [10]

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 4.3.7. és 4.3.8-as ábrán.



4.3.7. ábra – Bemeneti tagsági függvények levágása

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.



4.3.8. 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.

# Üzembe helyezés és kísérleti eredmények

## 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 terülén 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 “Református kollégium utca”-i kereszteződést 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 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)

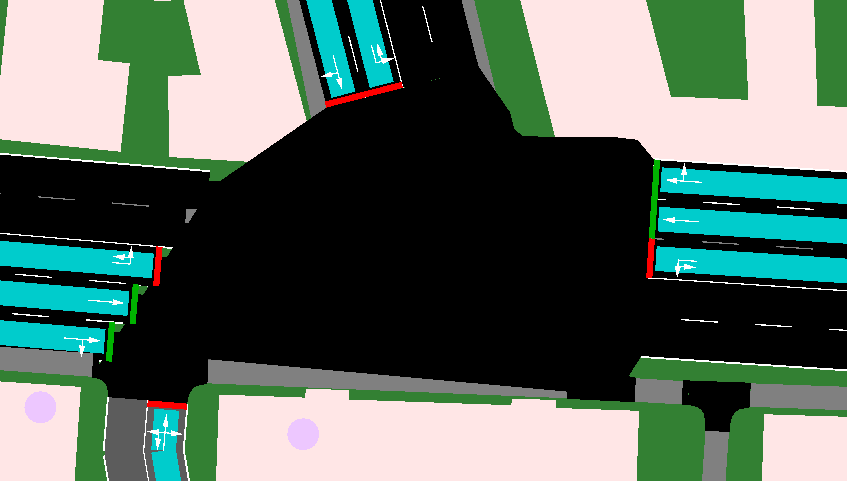
                        sequence\_time3-=delta\_t1

                next\_sequence+=sequence\_time1

                sequence+=1

5.1.2. ábra – Jelzőlámpák irányítása

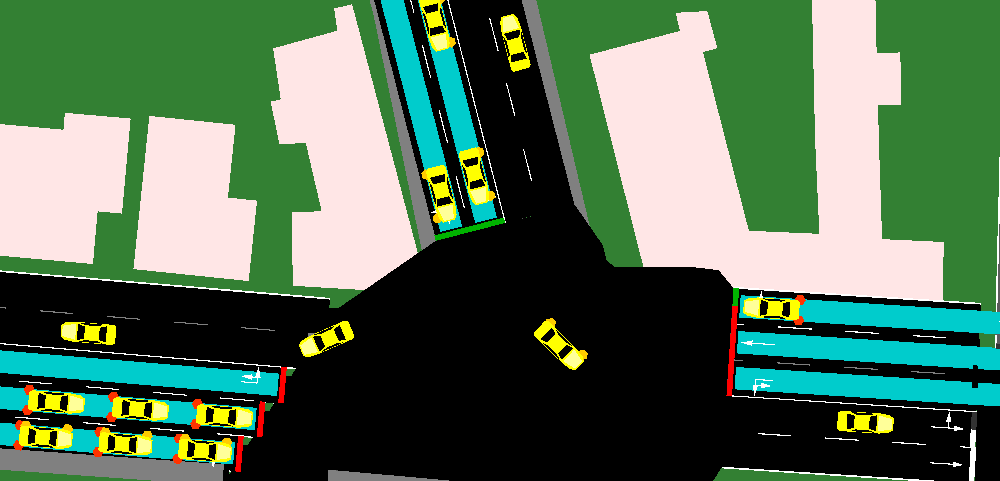
Ha a megfelelő szekvenciában vagyunk és a “step” változó elérte azt az időpillanatot, amikor az adott szekvencia ideje lejárt akkor a “traci” könyvtárban lévő “setRedYellowGreenState()” függvény segítségével módosítjuk a jelzőlámpákat a következő szekvenciának megfelelően. Ez a függvény úgy működik, 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 id/azonosító 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 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 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.

## Felmerült problémák és megoldásaik

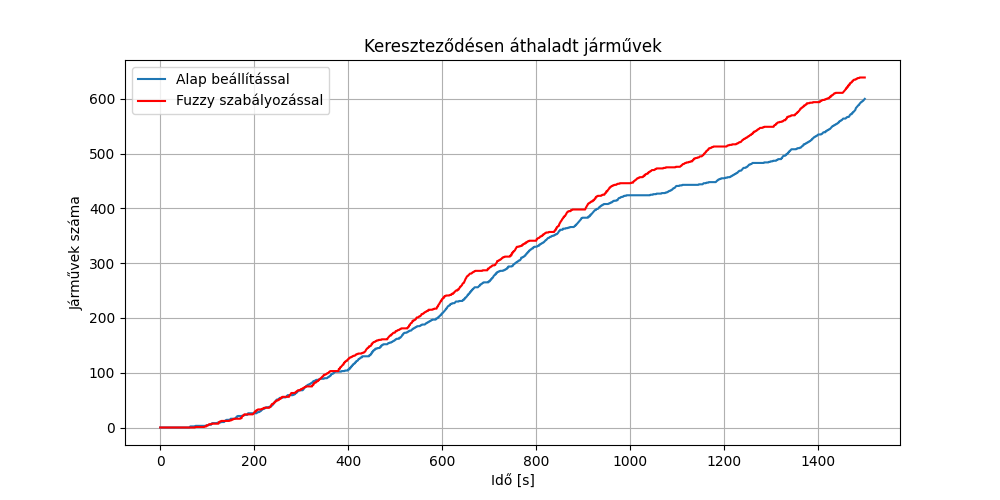
Egy kereszteződésben ahol négy útszakasz keresztezi egymást előfordult, hogy két útszakasz nagyon le volt terhelve és a másik két útszakaszon nagyon kevés autó közlekedik. Ilyenkor az a probléma merült fel, hogy a Fuzzy irányítás folyamatosan növeli azoknak a szekvenciáknak a hosszát, ahol sok jármű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

Ennek a problémának a megoldására bevezettünk egy minimális időt, amennyit minden szekvencia legalább kell hogy tartson. Így, ha a Fuzzy irányítás negatív értéket térít vissza akkor sem csökkentjük tovább a szekvencia hosszát. Viszont ebben az esetben is megtörtént, hogy 0 másodpercig 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 így meg tudjuk oldani ezt a problémát.

## 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. Ezt az értéket minden lépésben hozzáfűzzük egy listához és a lista elemeit meghatározott időnként kimentjük egy fájlba. 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.

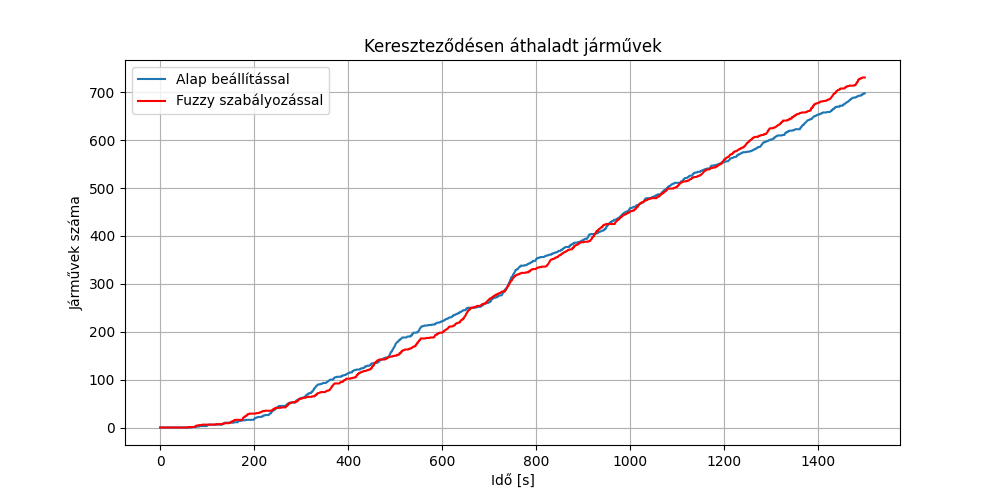
A következő ábrán láthatjuk a központban lévő kereszteződésben elvégzett mérést kirajzolva egy grafikonra az alap beállítással áthaladt autók számát és Fuzzy irányítással áthaladt autók számát idő függvényében. Minden szekvenciának a hossza kezdetben 25 másodperc. A szimulációt 1500 másodpercig futtatuk. Az elején kisebb forgalommal indul a szimuláció és fokozazosan növekszik. A 200-dik másodpercben már az 1000 autót is eléri a szimuláció és később a maximális forgalom esetén 4000-5000 autó közlekedik egész Marosvásárhely területén.



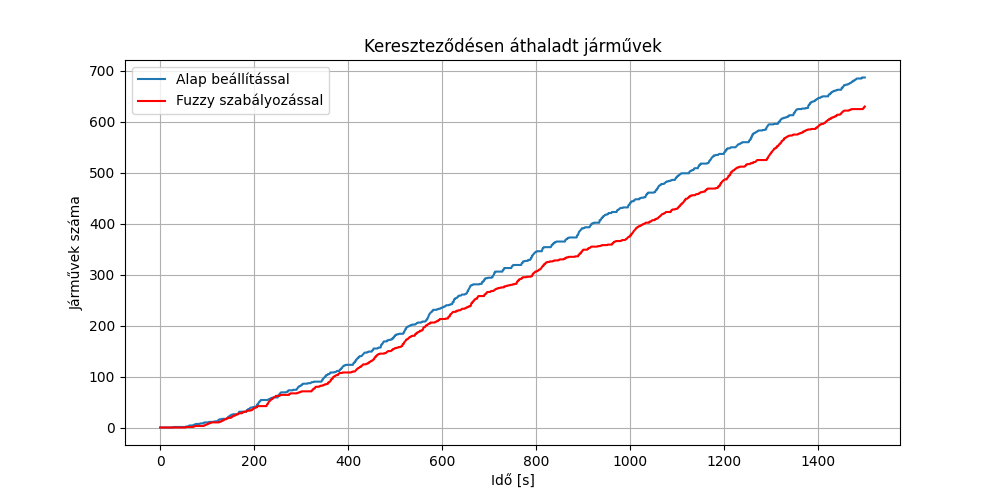
5.3.1. ábra – Központban áthaladt autók

Láthatjuk, hogy Fuzzy irányítással több autó halad át a kereszteződésen. Az elején 400 másdopercig, ahol még kisebb a forgalom hasonló számú autó halad át mindkét esetben, de ahogy növekszik a forgalom a Fuzzy szabályozással ellátott szimuláció jobban teljesít.

Ezt követően a “Református kollégium utca”-i kereszteződésnél végeztük a mérést. Ezt a mérést az 5.3.2-es ábra szemlélteti. Ebben az esetben három szekvenciára osztottuk a kereszteződést és ezért 30 másodpercnyi időt adtunk kezdetben minden szekvenciának.



5.3.2. ábra – Református kollégium utcai kereszteződésben áthaladt autók



5.3.3. ábra – Fortuna kereszteződésben áthaladt autók

# Következtetések

Ebben a fejezetben össze kell foglalni az elvégzett munka végén levont következtetéseket és tapasztalatokat az alábbi al-pontok szerint, de lehet más szempontokat is választani.

## Megvalósítások

## Hasonló rendszerekkel való összehasonlítás

## Továbbfejlesztési lehetőségek

Terjedelem: 1-2 oldal.

# 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. Javed Alam, Dr. M. K. Pandey, Husain Ahmed, “Development of Traffic Light Control System for Emergency Vehicle Using Fuzzy Logic”, 2013.
10. Tan Kok Khiang, Marzuki Khalid, Rubiyah Yusof, “Intelligent Traffic Lights Control By Fuzzy Logic”, 1997.

# Függelék

**A függelék tartalmazza a forráskódot és dokumentációt tartalmazó adathordozót (pl. CD), ezen kívül pedig bármilyen más kiegészítő anyagot ami nem fért be a dolgozatba. Ezekre hivatkozni kell a dolgozat szövegéből.**

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA

FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ

SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Vizat decan Vizat director departament

Conf. dr. ing. Domokos József Ș.l. dr. ing Szabó László Zsolt

1. https://trlsoftware.com/products/traffic-control/scoot/ [↑](#footnote-ref-1)
2. https://www.scats.nsw.gov.au/home [↑](#footnote-ref-2)
3. https://www.trafitek.com/atc-utopia [↑](#footnote-ref-3)
4. https://eclipse.dev/sumo/ [↑](#footnote-ref-4)
5. https://matsim.org/ [↑](#footnote-ref-5)
6. https://www.ptvgroup.com/en/products/ptv-vissim [↑](#footnote-ref-6)
7. https://www.aimsun.com/ [↑](#footnote-ref-7)
8. https://sumo.dlr.de/docs/index.html [↑](#footnote-ref-8)
9. https://www.openstreetmap.org/ [↑](#footnote-ref-9)