KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FUKULTETAS

Intelektikos pagrindai

(T120B029)

Laboratorinis darbas Nr. 3
Fuzzy – Miglotoji logika

Darbą atliko:
IFF-7/14 gr. Studentas
Eligijus Kiudys

Darbą priėmė:
lekt. Andrius Nečiūnas
doc. Agnė Paulauskaitė-Tarasevičienė

KAUNAS 2020

Turinys

Darbo užduoties aprašas, įrankių pasirinkimas 3
Užduotis 3
Įvestys bei išvestis 4
Įvestys 4
Išvestis 4
Detalesnė informacija apie pasirinktus rėžius 4
Taisyklių suformavimas 7
Formavimo taisyklės 7
Sudarytos taisyklės 7
Agregacija ir defuzifikacija 8
Testavimas su duomenų rinkiniu 9
Gautų rezultatų verifikavimas 11
Įrankio konfigūracija 11
Duomenų verifikavimas 14
Išvados 19
Programos kodas 20

Pav. 1 Atstumo suskirstymo grafikas 5
Pav. 2 Paros suskirstymo grafikas 5
Pav. 3 stotelių pasiskirstymo grafikas 6
Pav. 4 važiavimo trukmės pasiskirstymo grafikas 6
Pav. 5 pirmo testinio rinkinio agragacijos rezultatai. 9
Pav. 6 Antro testinio rinkinio agragacijos rezultatai. 9
Pav. 7 Trečio testinio rinkinio agragacijos rezultatai. 10
Pav. 8 Mamdami modelis 11
Pav. 9 MATLAB atstumo rėžiai ir suskirstymas 12
Pav. 10 MATLAB paros laiko rėžiai ir suskirstymas 12
Pav. 11 MATLAB stotelių kiekio rėžiai ir suskirstymas 13
Pav. 12 MATLAB važiavimo trukmės rėžiai ir suskirstymas 13
Pav. 13 MATLAB taisyklių sudarymas, pirma dailis 14
Pav. 14 MATLAB taisyklių sudarymas, antra dalis 14
Pav. 15 MATLAB rezultatai naudojant pirmą testinį rinkinį, centroid metodu 15
Pav. 16 MATLAB rezultatai naudojant pirmą testinį rinkinį, MOM metodu 15
Pav. 17 MATLAB rezultatai naudojant antrą testinį rinkinį, centroid metodu 16
Pav. 18 MATLAB rezultatai naudojant antrą testinį rinkinį, MOM metodu 17
Pav. 19 MATLAB rezultatai naudojant trečią testinį rinkinį, centroid metodu 17
Pav. 20 MATLAB rezultatai naudojant trečią testinį rinkinį, MOM metodu 18

Table. 1 MATLAB ir Python rezultatai bei paklaidos 16

Darbo užduoties aprašas, įrankių pasirinkimas

Užduotis

Sukurkite sprendimo priėmimo sistemą remiantis miglotosios logikos teorija (rekomenduojama taikant Mamdani algoritmą, tačiau gali būti naudojamas ir Sugeno modelis). Duomenys gali būti naudojami realūs, iš atvirų šaltinių arba sugalvoti jūsų pačių (dažniausiai studentai sugalvoja savo duomenis ir patiems aktualią problemą – t.y. jūs tampate ekspertais). Sistemos programinė realizacija turi būti atlikta naudojant Matlab arba Python

Užduoties sprendimui yra naudojamas Mamdani algoritmas, kuris bus taikomas naudojant Python programavimo kalbą.

Naudojamos Python bibliotekos laboratoriniame darbe:

1. Numpy – darbui su masyvais ir papildomiems skaičiavimams.
2. Matplotlib – grafikų braižymui.

Šiame laboratoriniame darbe pasirinkau prognozuoti autobuso arba troleibuso važiavimo trukmę nuo vienos pasirinktos stotelės iki kitos pasirinktos stotelės. Prognozavimui pasirinkau tris Įvestis: maršruto atstumas, paros metas, tarpinių stotelių skaičius.

Įvestys bei išvestis

Pasirinkta prognozuojama reikšmė - Važiavimo trukmė.

Įvestys

Naudojami duomenys yra išgalvoti.
Įvesčių ir atsakymo rėžiai:

1. Maršruto atstumas – atstumas kilometrais kuris yra skaičiuojamas nuo pirmos pasiriktos stotelės iki galutinės stotelės. Rėžiai yra nuo vieno iki trisdešimt kilometrų. Atstumas yra suskirstytas į tris dalis: mažas, vidutinis, didelis.
2. Paros metas – autobuso važiavimo paros metas. Rėžiai yra nuo nulio iki dvidešimt trijų valandų. Paros metas taip pat yra suskirstytas į keturias dalis: naktis, rytas, diena, vakaras.
3. Stotelių kiekis – kiekis kuris nurodo kiek yra tarpinių stotelių tarp pasirinktų pirmos ir paskutinės stotelių. Rėžiai yra nuo vieno iki dvidešimt. Stotelių kiekis yra suskirstytas į tris dalis: mažas, vidutinis, didelis.

Išvestis

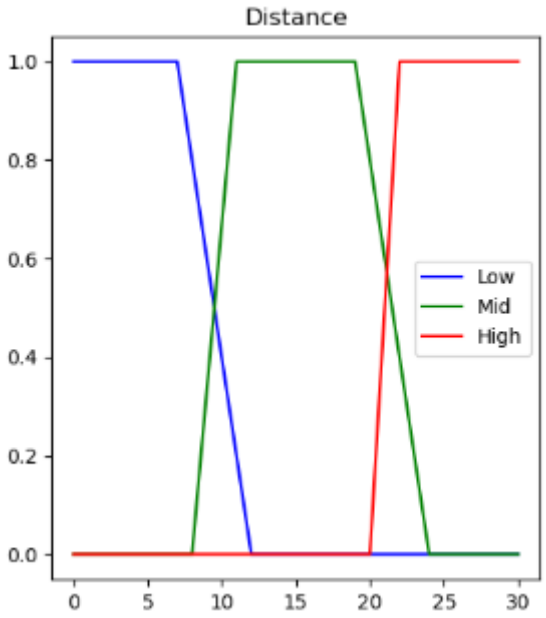
Važiavimo trukmė – trukmė nurodanti kiek laiko užtruks atvažioti nuo vienos pasirinktos iki kitos pasirinktis stotelės minutėmis. Rėžiai yra nuo vienos iki šešiasdešimt penkių minučių. Važiavimo trukmė naudojimui yra suskirstyta į tris dalis: mažas, vidutinis, didelis.

Detalesnė informacija apie pasirinktus rėžius

Kiekviena įvestis ir išvestis yra skirstoma bent į tris dalis (mano pasirinkimu). Duomenys yra skirstomi į pasirinktą dalių kiekį taisyklių sudarymui ir prognozavimui. Gerai suskirstyti duomenys lemia prognozavimo tikslumą. Suskirstymas yra išgalvotas todėl mano nuomonė su jūsų gali skirtis.

Maršruto atstumas yra skirstomas į tris dalis kurios turi savo rėžius. Pasirinkti rėžiai gali keistis nuo pasirinkto miesto ar šalies.

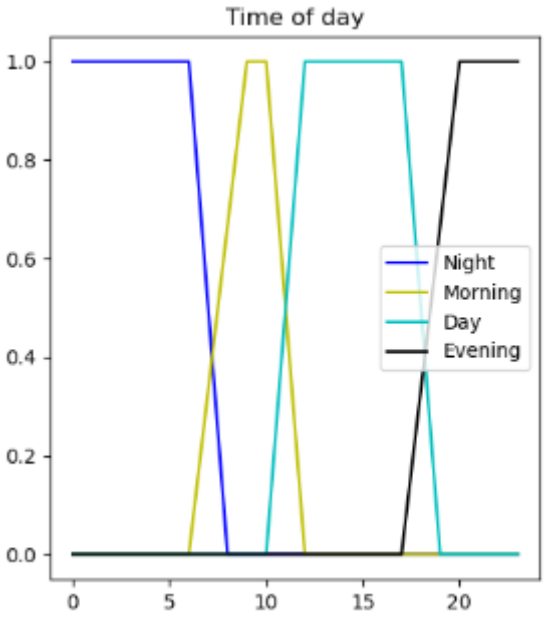
1. Mažas – Nuo nulio iki dvylikos minučių.
2. Vidutinis – Nuo aštuonių iki dvidešimt keturių minučių.
3. Didelis – Nuo dvidešimt iki trisdešimt minučių.



Pav. 1 Atstumo suskirstymo grafikas

Paros metas yra skirstomas į keturias dalis, kurios turi nurodytus rėžius.

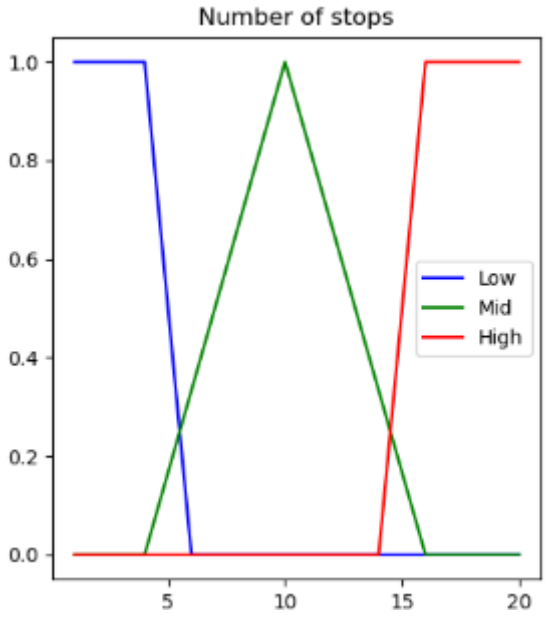
1. Naktis – Nuo nulio iki aštuntos valandos ryto
2. Rytas – Nuo šeštos valandos ryto iki dvyliktos valandos.
3. Diena – Nuo dešimtos valandos iki devynioliktos valandos.
4. Vakaras -Nuo septynioliktos valandos iki dvidešimt trečios valandos.



Pav. 2 Paros suskirstymo grafikas

Tarpinis stotelių kiekis skirstomas į tris dalis kurios persidengia.

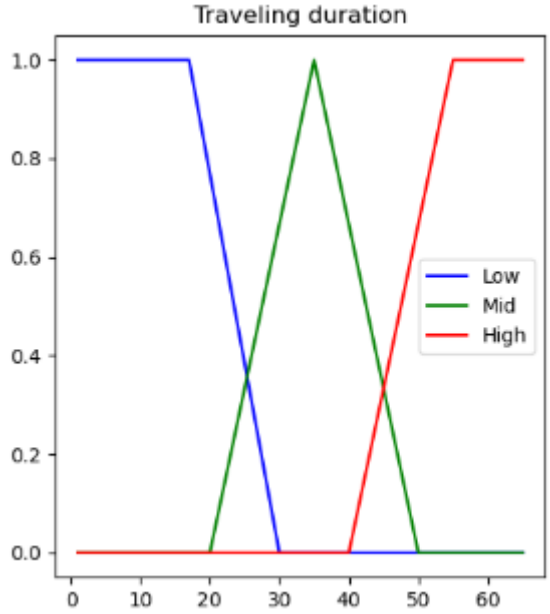
1. Mažas – mažas stotelių kiekis yra nuo vienos iki šešių stotelių.
2. Vidutinis – vidutinis kiekis yra nuo keturių iki šešiolika stotelių.
3. Didelis – didelis kiekis yra nuo keturiolikos iki dvidešimt stotelių.



Pav. 3 stotelių pasiskirstymo grafikas

Važiavimo trukmė taip pat skirstoma į tris dalis kurios persidengia.

1. Mažas – mažas važiavimo laikas yra nuo vienos iki trisdešimt minučių.
2. Vidutinis – vidutinis laikas yra nuo dvidešimt iki penkiasdešimt minučių.
3. Didelis – didelis laikas yra nuo keturiasdešimt iki šešiasdešimt penkių minučių.



Pav. 4 važiavimo trukmės pasiskirstymo grafikas

Taisyklių suformavimas

Formavimo taisyklės

Taisyklių suformavimui naudoju ir, arba (Angl. and, or) loginius kintamuosius.

Taisyklė formuluojama naudojant pagal pateiktus formatus.

1. Jei S1 arba S2 arba … arba Sn tai išvada.
2. Jei S1 arba S2 tai išvada.
3. Jei S1 ir S2 ir … ir Sn tai išvada.
4. Jei S1 ir S2 tai išvada.

Taisyklėje naudojant žodį ir, tarp duotų koeficientų bus ieškomas minimumas. Sąlygoje naudojant žodį arba bus ieškoma tarp duotų koeficientų bus ieškomas maksimumas.

Sudarytos taisyklės

1. Jeigu atstumas yra didelis arba tarpinių stotelių kiekis didelis tai važiavimo trukmė yra didelė.
2. Jeigu atstumas yra mažas ir paros metas yra naktis tai važiavimo trukmė yra maža.
3. Jeigu atstumas yra vidutinis ir arba tarpinių stotelių kiekis yra didelis tai važiavimo trukmė yra didelė.
4. Jeigu atstumas yra mažas ir tarpinių stotelių kiekis yra mažas tai važiavimo trukmė yra maža.
5. Jeigu atstumas yra vidutinis arba tarpinių stotelių kiekis yra vidutinis tai važiavimo trukmė yra vidutinė.
6. Jeigu paros metas yra rytas arba tarpinių stotelių kiekis didelis tai važiavimo trukmė yra didelė.
7. Jeigu atstumas yra didelis ir paros metas yra diena tai važiavimo trukmė yra vidutinė.
8. Jeigu atstumas yra vidutinis ir tarpinių stotelių kiekis didelis tai važiavimo trukmė yra didelė.
9. Jeigu tarpinių stotelių kiekis didelis arba paros metas yra naktis tai važiavimo trukmė yra maža.
10. Jeigu atstumas yra vidutinis ir tarpinių stotelių kiekis yra mažas didelis tai važiavimo trukmė yra vidutinė.
11. Jeigu atstumas yra didelis ir tarpinių stotelių kiekis yra vidutinis tai važiavimo trukmė yra didelė.
12. Jeigu atstumas yra mažas ir paros metas yra diena tai važiavimo trukmė yra maža.
13. Jeigu atstumas yra mažas ir tarpinių stotelių kiekis yra vidutinis tai važiavimo trukmė yra vidutinė.

Agregacija ir defuzifikacija

Agregacijai pasirinkau max metodą. Naudojant max metodą yra išrenkamos suskirstytos taisyklės su didžiausia reikšme. Agregacija suteikia galimybę naudoti visas norimas taisykles.

Taisykles suskirsčiau į tris dalis mažą, vidutinę ir didelę. Išrinkus maksimumus iš suskirstytų reikšmių naudojau galutinį agregavimo žingsnį y maksimumo paieška.

Įvykdžius agregavimą yra naudojami du pasirinkti defuzifikacijos metodai, centroid ir MOM (Angl. Mean of maximum).

Centroid metodas kitaip vadinamas COG arba COA skaičiuoja ploto centrą naudojant $y = \frac{\int x\ \mu_A(x)dx}{\int \mu_A(x)dx}$ formulę arba $y = \frac{\Sigma_{i=1}^{n} x_i \times \mu(x_i)}{\Sigma_{i=1}^{n} \mu(x_i)}$ formulę.

MOM metodas dar kitaip vadinamas maksimumų vidurkio metodas. Skaičiavimui naudojamos defuzifikuotos y reikšmės kurios yra didžiausios. Naudojant šitas reikšmes surandamos x reikšmės ir apskaičiuojamas x reikšmių vidurkis. Naudojama formulė:

$y_0 = mean\{y: m(y) = \max\{m(y)\}\}$

Testavimas su duomenų rinkiniu

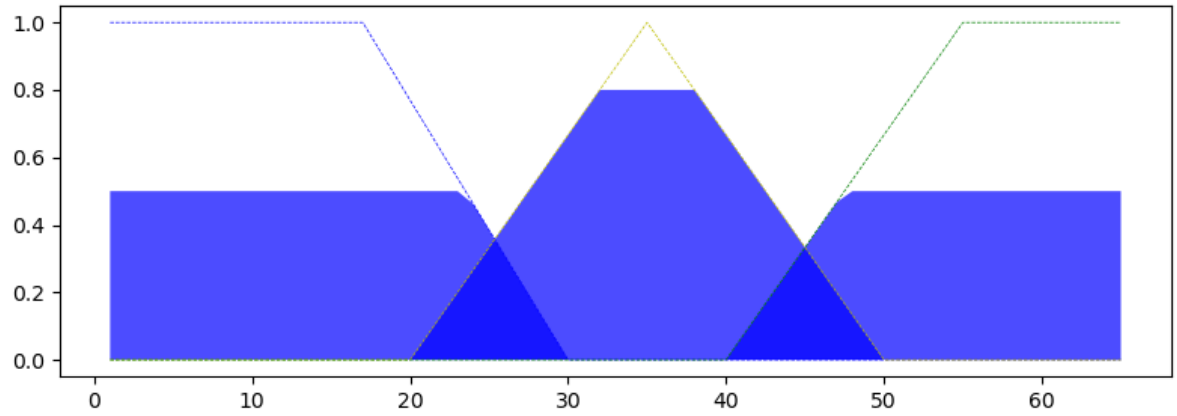
Testavimui naudoju tris skirtingus duomenų rinkinius.

Pirmasis duomenų rinkinys:

- Maršruto atstumas – 20, vidutinis
- Paros metas – 7 valanda, naktis, rytas.
- Stotelių kiekis- 15, vidutinis, didelis.

Išvestys:

- Važiavimo trukmė naudojant centroid metoda: 33.09 min
- Važiavimo trukmė naudojant MOM metoda: 35.0 min



Pav. 5 pirmo testinio rinkinio agragacijos rezultatai.

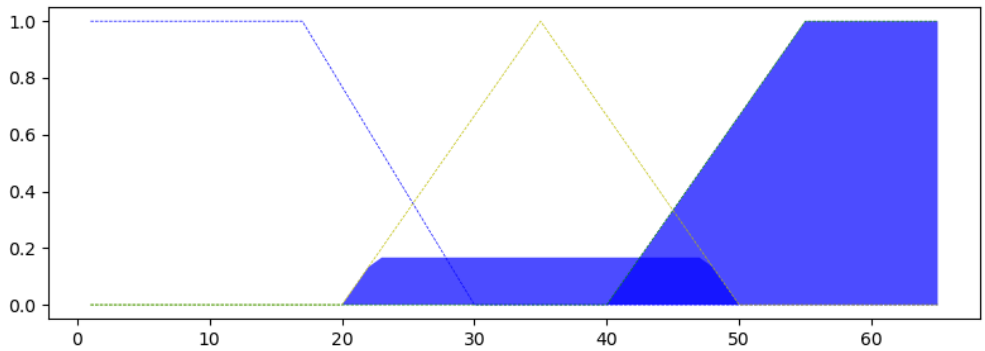
Pasiremdami grafiku matome jog įvestys labiausiai taikėsi į vidutinį važiavimo trukmės laiką, ką atspindi ir gauti rezultatai naudojantis centroid ir MOM metodus. Tačiau matome kad mažo laiko ir didelio laiko dalys taip pat yra užpildytos.

Antrasis Duomenų rinkinys:

- Maršruto atstumas – 4 km, vidutinis
- Paros metas – 9 valanda, rytas.
- Stotelių kiekis- 15, vidutinis, didelis.

Išvestys:

- Važiavimo trukmė naudojant centroid metodą: 51.8 min
- Važiavimo trukmė naudojant MOM metodą: 60.0 min



Pav. 6 Antro testinio rinkinio agragacijos rezultatai.

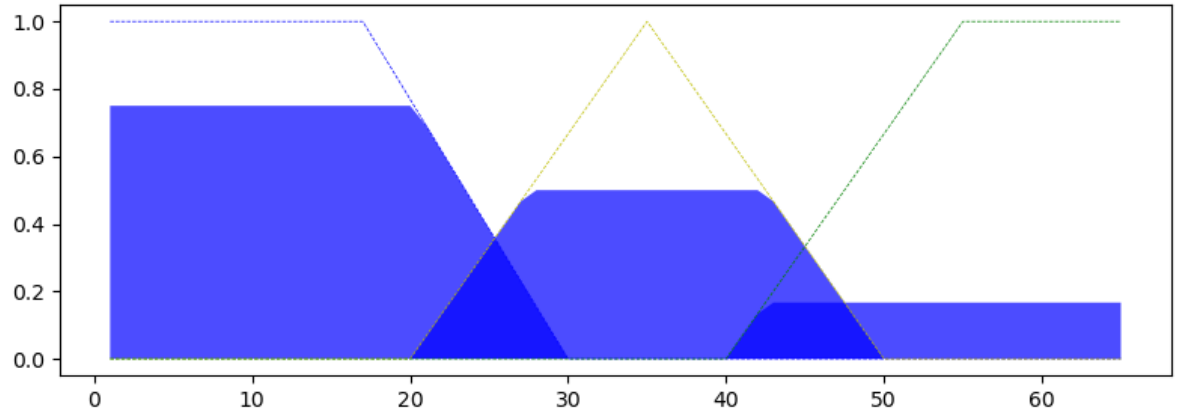
Remiantis grafiku matome jog įvestys taikosi į didžiausią važiavimo trukmę. Atsižvelgiant į įvestus duomenis rezultatą atvaizduoja ganėtinai tiksliai. Matome kad yra taikytasi ir į vidutinį laiką, bet vidutinio laiko srities užpildymas yra mažas.

Trečiasis Duomenų rinkinys:

- Maršruto atstumas – 5 km, vidutinis.
- Paros metas – 6.5 valanda, rytas, naktis
- Stotelių kiekis- 13, vidutinis, didelis.

Išvestys:

- Važiavimo trukmė naudojant centroid metodą: 24.52 min
- Važiavimo trukmė naudojant MOM metodą: 10.5 min



Pav. 7 Trečio testinio rinkinio agragacijos rezultatai.

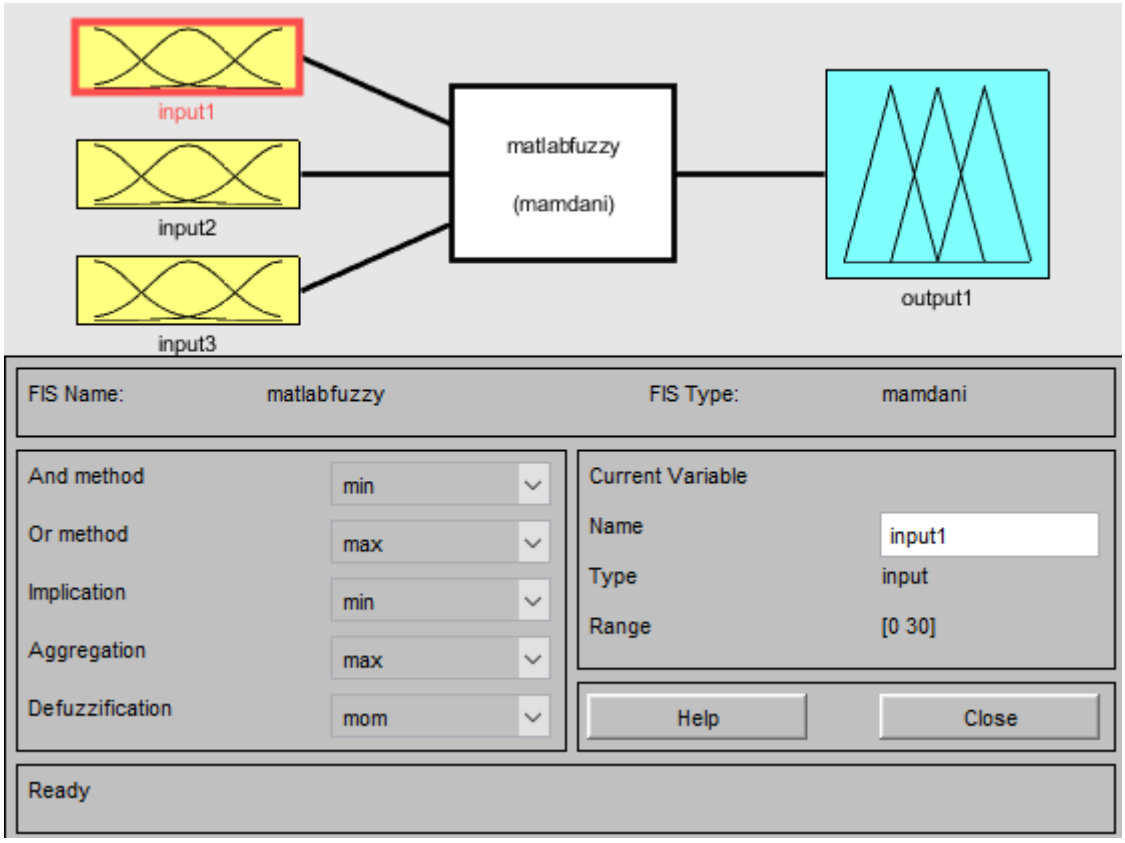
Grafike matome jog trukmės užpildymas yra visur, bet daugiausia užpildo mažos trukmės pasiskirstymą, taip pat matome, kad centroido trukmė skiriasi daug nuo mom trukmės, kadangi mom skaičiuoja aukčiausio y, x vidurkį, o centroidas skaičiuoja viso ploto vidurį.

Gautų rezultatų verifikavimas

Įrankio konfigūracija

Rezultatų verifikavimui naudoju MATLAB Fuzzy Logic Designer Toolbox įrankį.

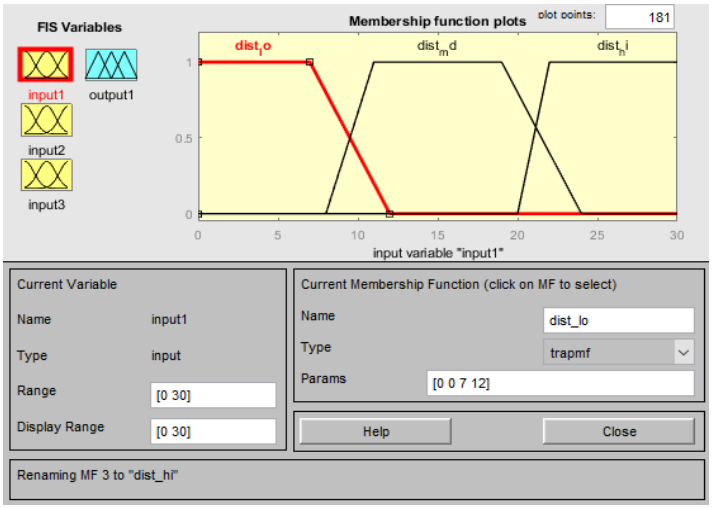
Pradžioje susikūriau mamdani naują modelį, tada prisidėjau dar du papildomus inputus ir pasirinkau metodus: and, or, implication, aggregation ir defuzzification.



Pav. 8 Mamdami modelis

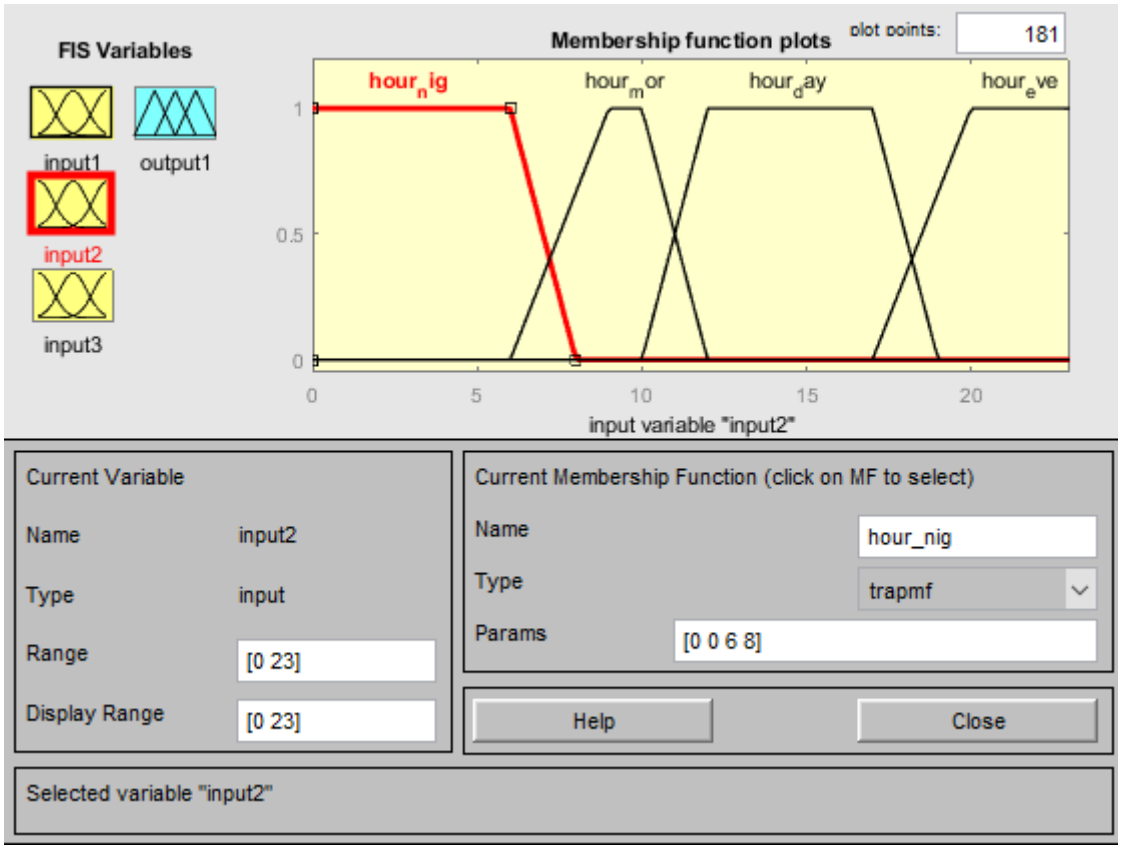
Susidaręs modelį pradėjau keisti įvesties rėžius, grafiko suskirstymo kiekį ir pavadinimus

Atstumo grafikas.



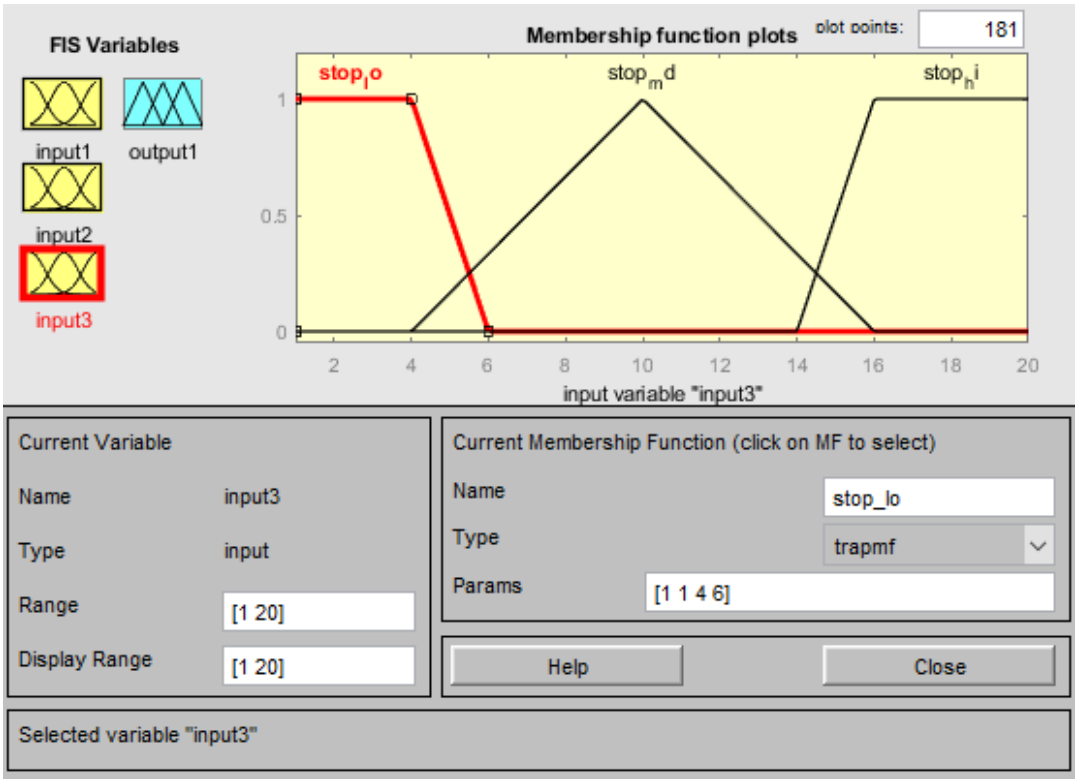
Pav. 9 MATLAB atstumo rėžiai ir suskirstymas

Paros laiko grafikas.



Pav. 10 MATLAB paros laiko rėžiai ir suskirstymas

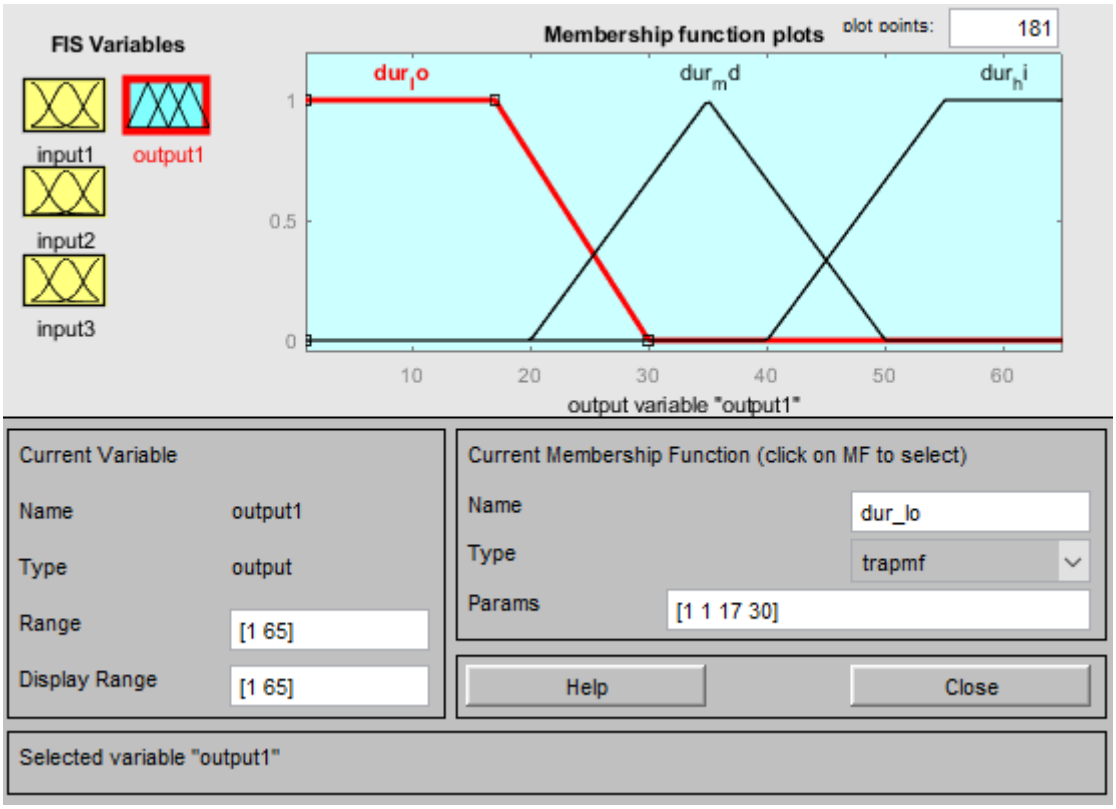
Stotelių kiekis.



Pav. 11 MATLAB stotelių kiekio rėžiai ir suskirstymas

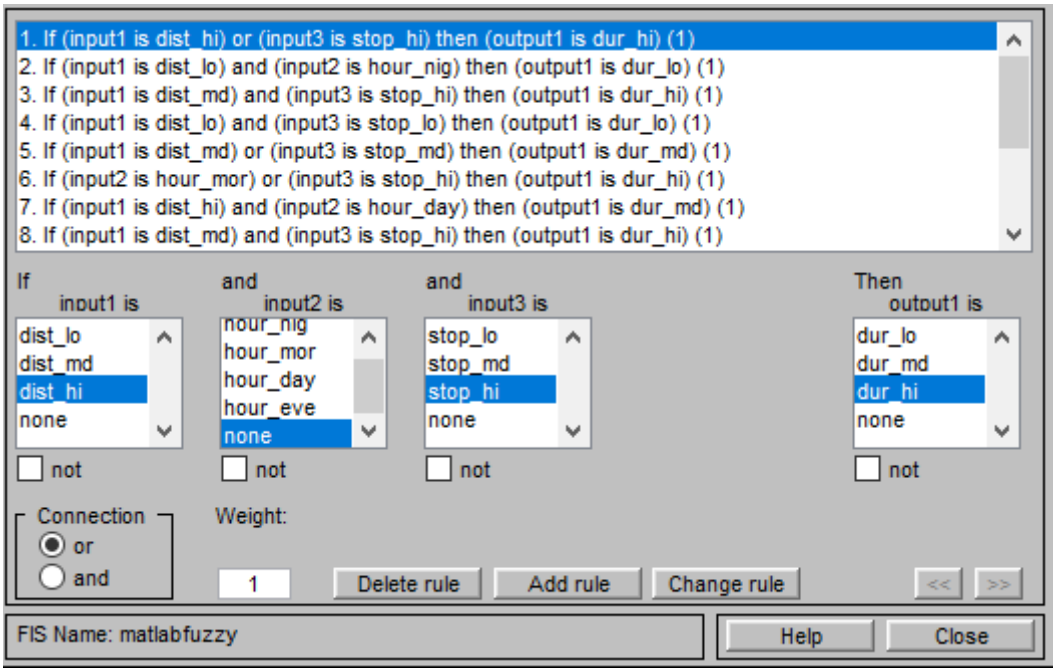
Susikonfigūraveęs visus įvesties rėžius ir paskirstymus atėjo metas suskirstyti rezultatą ir sutvarkyti rezultato rėžius.

Važiavimo trukmė.

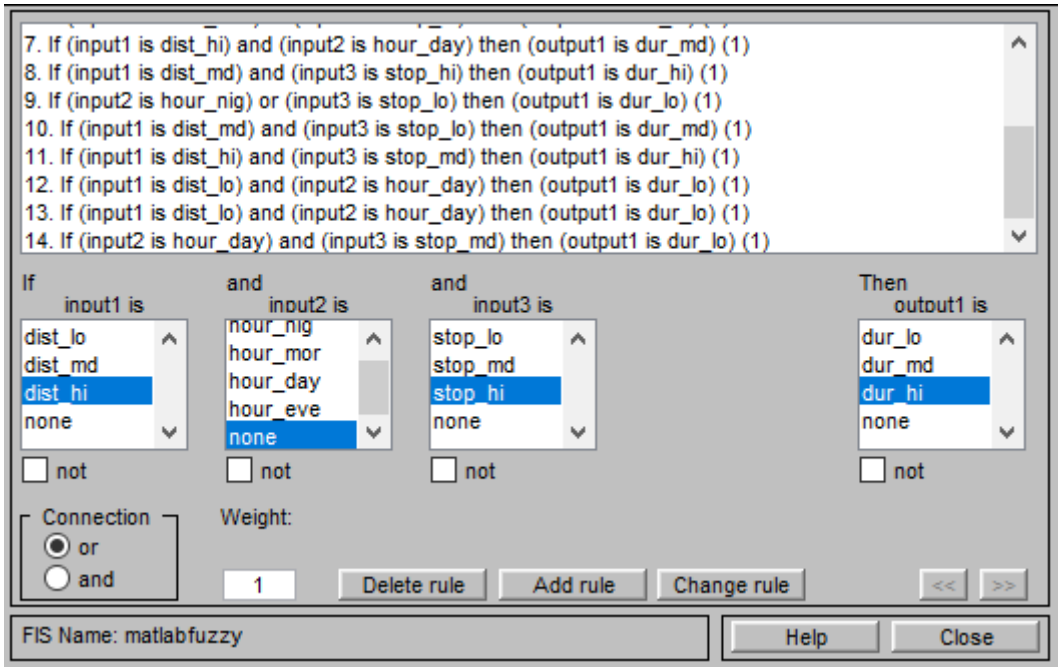


Pav. 12 MATLAB važiavimo trukmės rėžiai ir suskirstymas

Susidaręs visus grafikus galime susikurti visas taisykles naudojamame įrankyje.



Pav. 13 MATLAB taisyklių sudarymas, pirma dailis



Pav. 14 MATLAB taisyklių sudarymas, antra dalis

Duomenų verifikavimas

Testavimui naudoju tris duomenų rinkinius kuriuos naudojau ir python prognozavime.

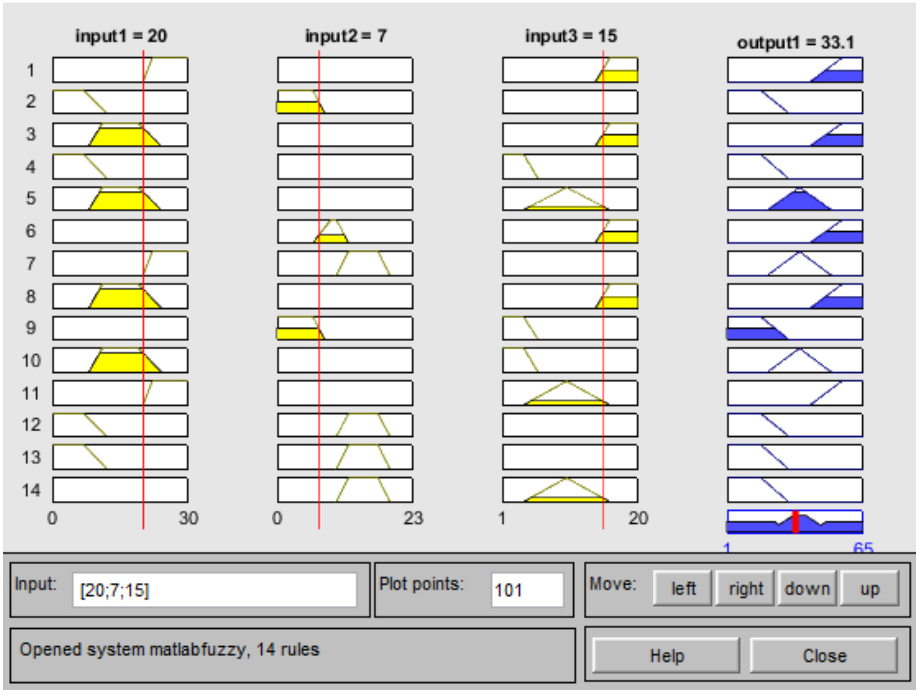
Pirmasis duomenų rinkinys:

- Maršruto atstumas – 20, vidutinis
- Paros metas – 7 valanda, naktis, rytas.
- Stotelių kiekis- 15, vidutinis, didelis.

Išvestys naudojant Python:

- Važiavimo trukmė naudojant centroid metoda: 33.09 min
- Važiavimo trukmė naudojant MOM metodą: 35.0 min

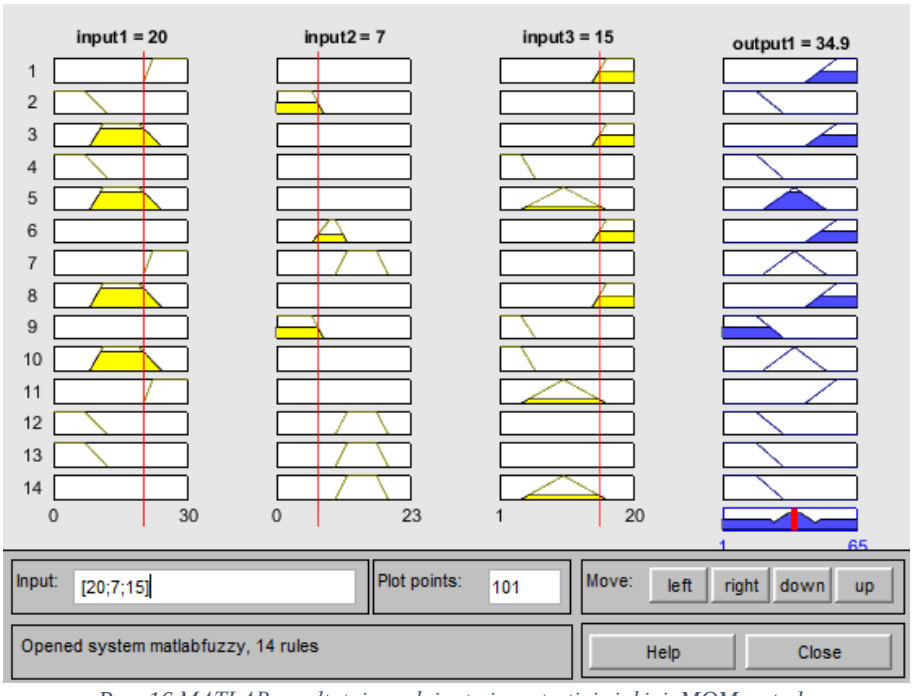
MATLABO grafinis sprendimas.
Pasirinkus centroid metodą ir įvedus įvedimus 20, 7 ir 15 gauname tokį grafinį spendimą.



Pav. 15 MATLAB rezultatai naudojant pirmą testinį rinkinį, centroid metodu

Palyginus mano gautą rezultatą 33.09 minutės ir MATLAB įrankio rezultatas 33.1 kuris parodo, kad paklaida yra apie 0.01 minutės. Gauta paklaida yra labai maža.

Naudojant pirmą tęstinį rinkinį su MOM defuzifikacijos metodu gauname tokį grafinį sprendimą.



Pav. 16 MATLAB rezultatai naudojant pirmą testinį rinkinį, MOM metodu

Palyginus mano gautą rezultatą 35.0 minutės ir MATLAB įrankio rezultatas 34.9 kuris parodo, kad paklaida yra apie 0.1 minutę. Gauta paklaida yra maža.

Visus rezultatus ir paklaidas pateiksiu lentelėje ir po lentelės sukelsiu MATLAB grafinius rezultatus.

Maršruto atstumas kilometrais	Paros metas valandomis	Stotelių kiekis	Python Centroid	Python MOM	Matlab Centroid	Matlab MOM	Centroid paklaida	MOM paklaida
20	7	15	33.09	35.00	33.10	34.90	0.01	0.10
4	9	15	51.80	60.00	52.00	60.20	0.20	0.20
5	6.5	13	24.52	10.5	24.4	10.60	0.12	0.10

Table. 1 MATLAB ir Python rezultatai bei paklaidos

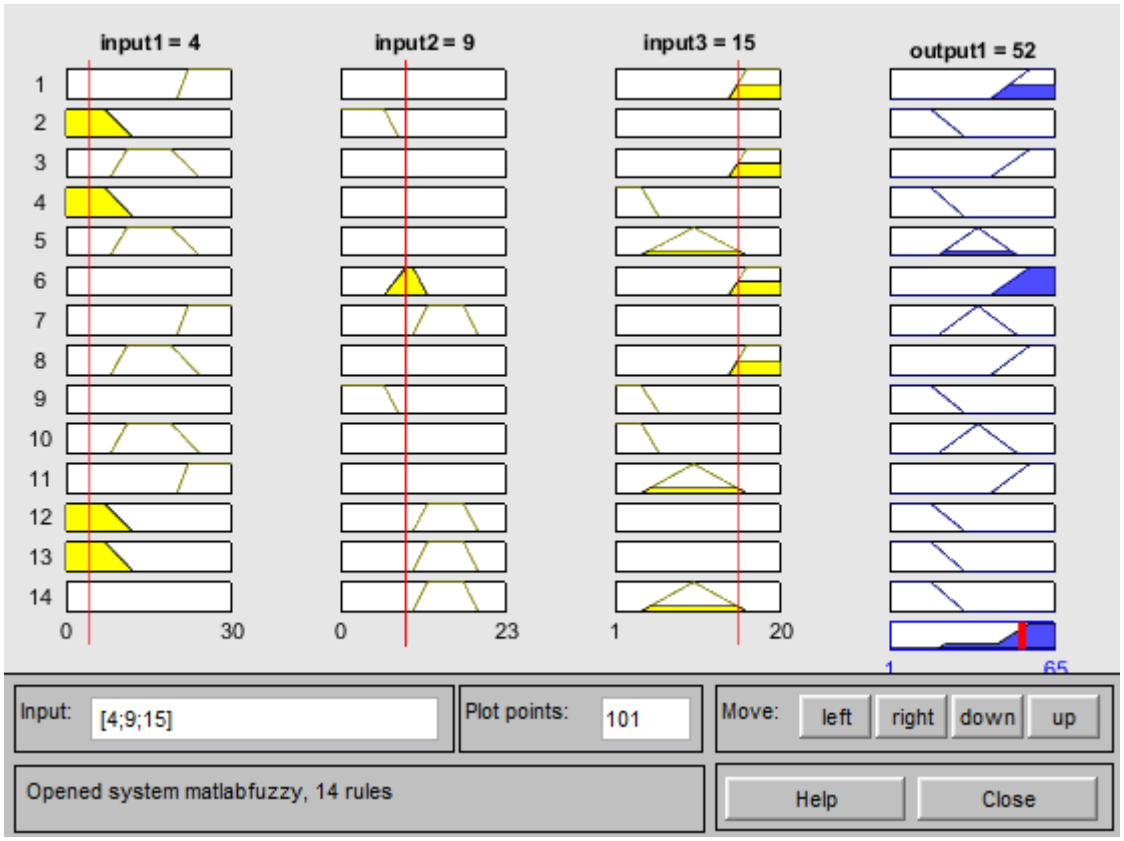
Antrasis Duomenų rinkinys:

- Maršruto atstumas – 4 km, vidutinis
- Paros metas – 9 valanda, rytas.
- Stotelių kiekis- 15, vidutinis, didelis.

Išvestys naudojant Python:

- Važiavimo trukmė naudojant centroid metodą: 51.8 min
- Važiavimo trukmė naudojant MOM metodą: 60.0 min

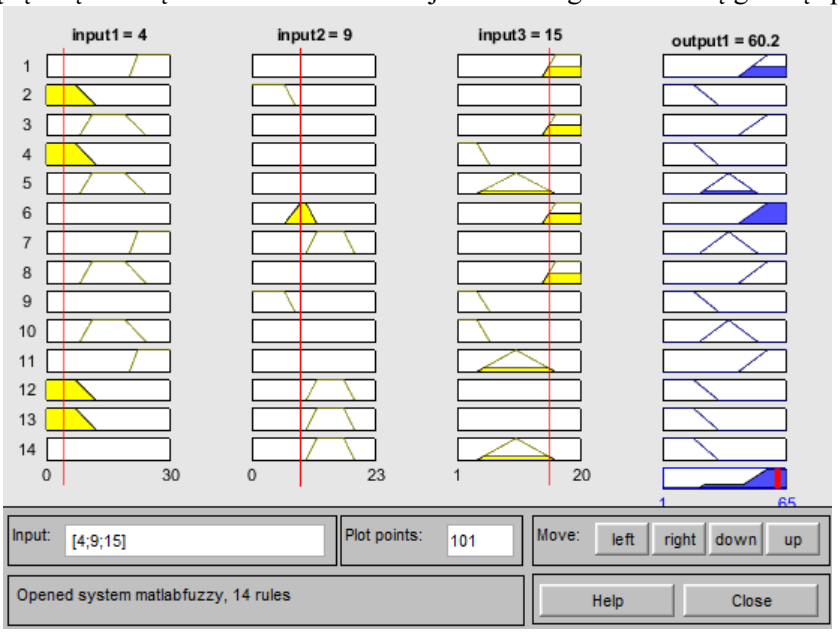
Pasirinkus centroid metodą ir įvedus įvestis 20, 7 ir 15 gauname tokį grafinį spendimą.



Pav. 17 MATLAB rezultatai naudojant antrą testinį rinkinį, centroid metodu

Palyginus mano gautą rezultatą naudojant Python 51.08 minutės ir MATLAB įrankio rezultatas 52.0 kuris parodo, kad paklaida yra apie 0.2 minutės. Gauta paklaida yra maža.

Naudojant antrą tęstinį rinkinį su MOM defuzifikacijos metodu gauname tokį grafinį sprendimą.



Pav. 18 MATLAB rezultatai naudojant antrą testinį rinkinį, MOM metodu

Palyginus mano gautą rezultatą 60.0 minutės ir MATLAB įrankio rezultatas 60.2 kuris parodo, kad paklaida yra apie 0.2 minutės. Gauta paklaida yra maža.

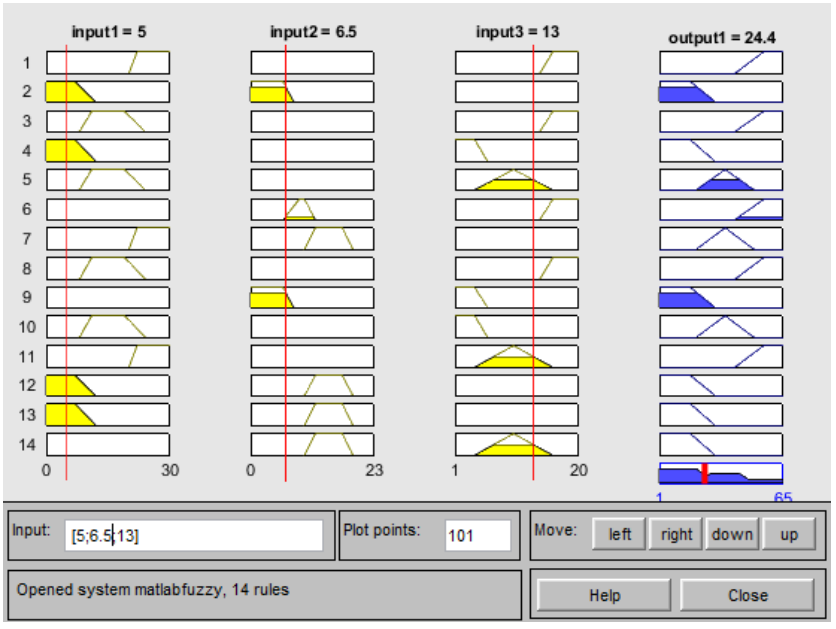
Trečiasis Duomenų rinkinys:

- Maršruto atstumas – 5 km, vidutinis.
- Paros metas – 6.5 valanda, rytas, naktis
- Stotelių kiekis- 13, vidutinis, didelis.

Išvestys naudojant Python:

- Važiavimo trukmė naudojant centroid metodą: 24.52 min
- Važiavimo trukmė naudojant MOM metodą: 10.5 min

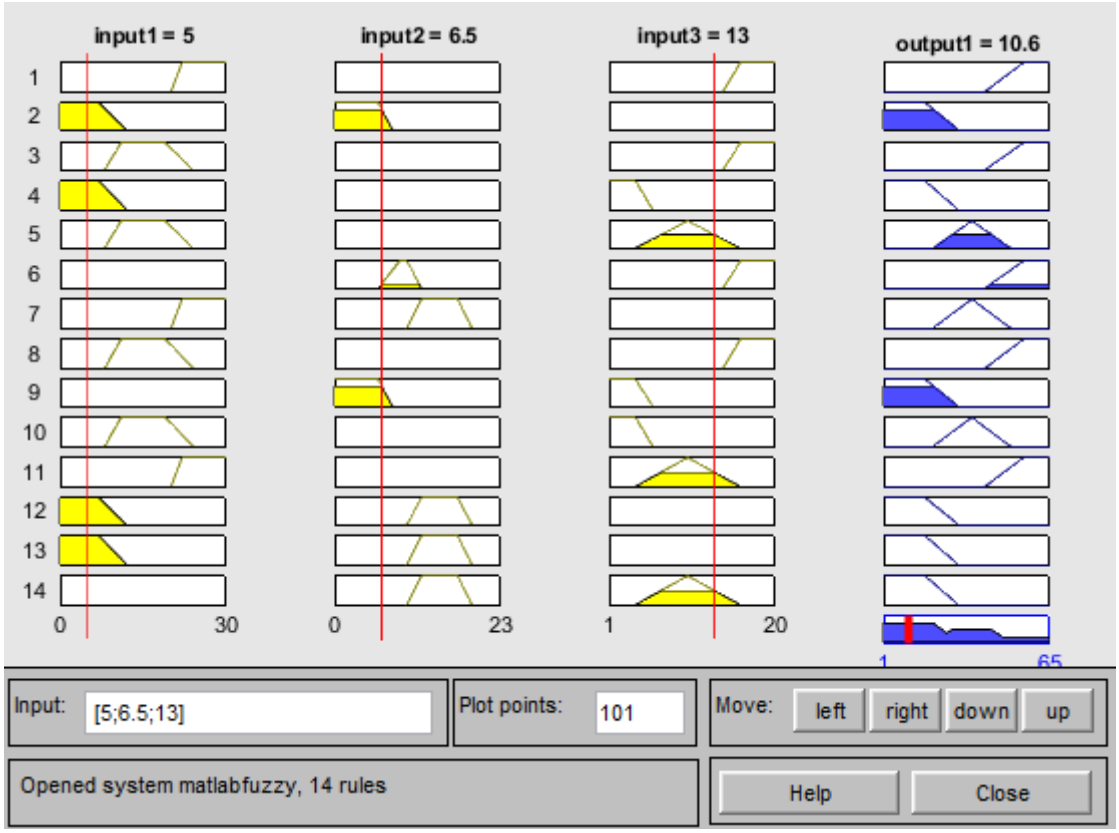
Pasirinkus centroid metodą ir įvedus įvestis 5, 6.5 ir 13 gauname tokį grafinį spendimą.



Pav. 19 MATLAB rezultatai naudojant trečią testinį rinkinį, centroid metodu

Palyginus mano gautą rezultatą naudojant Python 24.52 minutės ir MATLAB įrankio rezultatas 24.40 kuris parodo, kad paklaida yra apie 0.20 minutės. Gauta paklaida yra maža.

Naudojant trečią tęstinį rinkinį su MOM defuzifikacijos metodu gauname tokį grafinį sprendimą.



Pav. 20 MATLAB rezultatai naudojant trečią testinį rinkinį, MOM metodu

Palyginus mano gautą rezultatą naudojant Python 10.50 minutės ir MATLAB įrankio rezultatas 10.60 kuris parodo, kad paklaida yra apie 0.10 minutės. Gauta paklaida yra maža.

Išvados

Supratau kaip vyksta fuzzy moglotosios logikos skaičiavimai. Praktiškai pavyko realizuoti fuzzy miglotosios logikos skaičiavimus naudojant išgalvotą problemą ir juos duomenis. Duomenis verifikavau MATLAB fuzzy įrankiu, naudojantis įrankiu pastebėjau koks jis parastas ir galingas. Palyginus mano gautus rezultatus ir MATLAB įrankio, rezultatai mažai skyrėsi, todėl galiu teigti kad programą gerai realizavau. Naudojantis fuzzy miglotosios logikos algoritmais pamačiau kaip galima lengvai nuspėti atsakymą su gerai aprašytomis taisyklėmis. Fuzzy logika efektyviau sprendžia specifinio tipo užduotis nei dirbtinis intelektas, bet rezultatas gali skirtis. Mano atveju buvo galim sukurti daugiau taisyklių ir geriau paskirstyti rėžius norint gauti geresnį rezultatą.

Programos kodas

```
import numpy as np
import matplotlib.pyplot as plt


def trapecia(x, abcd):

    assert len(abcd) == 4, 'abcd parameter must have exactly four elements.'
    a, b, c, d = np.r_[abcd]
    assert a <= b and b <= c and c <= d, 'abcd requires the four elements \
                                         a <= b <= c <= d.'
    y = np.ones(len(x))

    idx = np.nonzero(x <= b)[0]
    y[idx] = triangle(x[idx], np.r_[a, b, b])

    idx = np.nonzero(x >= c)[0]
    y[idx] = triangle(x[idx], np.r_[c, c, d])

    idx = np.nonzero(x < a)[0]
    y[idx] = np.zeros(len(idx))

    idx = np.nonzero(x > d)[0]
    y[idx] = np.zeros(len(idx))

    return y


def triangle(x, abc):

    assert len(abc) == 3, 'abc parameter must have exactly three elements.'
    a, b, c = np.r_[abc]     # Zero-indexing in Python
    assert a <= b and b <= c, 'abc requires the three elements a <= b <= c.'

    y = np.zeros(len(x))

    # Left side
    if a != b:
        idx = np.nonzero(np.logical_and(a < x, x < b))[0]
        y[idx] = (x[idx] - a) / float(b - a)

    # Right side
    if b != c:
        idx = np.nonzero(np.logical_and(b < x, x < c))[0]
        y[idx] = (c - x[idx]) / float(c - b)

    idx = np.nonzero(x == b)
    y[idx] = 1
    return y


def pointPosTri(x, a, b, c):
    if(x < a):
        return 0
    elif(a <= x < b):
        return (x-a)/(b-a)
    elif(b <= x <= c):
        return (c-x)/(c-b)
```

```
    elif(x > c):
        return 0


def ProintPosTrap(x, a, b, c, d):
    if (x < a):
        return 0
    elif (a <= x < b):
        return (x - a) / (b - a)
    elif (b <= x < c):
        return 1
    elif(c <= x < d):
        return (d - x) / (d - c)
    elif(d <= x):
        return 0

def MOM(x, mfx):
    val = mfx == mfx.max()
    return np.mean(x[val])


def centroid(x, mfx):

    sum_moment_area = 0.0
    sum_area = 0.0

    # If the membership function is a singleton fuzzy set:
    if len(x) == 1:
        return x[0]*mfx[0] / np.fmax(mfx[0], np.finfo(float).eps).astype(float)

    # else return the sum of moment*area/sum of area
    # checking for triangles and square
    for i in range(1, len(x)):
        x1 = x[i - 1]
        x2 = x[i]
        y1 = mfx[i - 1]
        y2 = mfx[i]

        # if y1 == y2 == 0.0 or x1==x2: --> rectangle of zero height or width
        if not(y1 == y2 == 0.0 or x1 == x2):
            if y1 == y2:  # rectangle
                moment = 0.5 * (x1 + x2)
                area = (x2 - x1) * y1
            elif y1 == 0.0 and y2 != 0.0:  # triangle, height y2
                moment = 2.0 / 3.0 * (x2-x1) + x1
                area = 0.5 * (x2 - x1) * y2
            elif y2 == 0.0 and y1 != 0.0:  # triangle, height y1
                moment = 1.0 / 3.0 * (x2 - x1) + x1
                area = 0.5 * (x2 - x1) * y1
            else:
                moment = (2.0 / 3.0 * (x2-x1) * (y2 + 0.5*y1)) / (y1+y2) + x1
                area = 0.5 * (x2 - x1) * (y1 + y2)

            sum_moment_area += moment * area
            sum_area += area

    return sum_moment_area / np.fmax(sum_area, np.finfo(float).eps).astype(float)
```

```
x_dist = np.arange(0, 31, 1)
x_hour = np.arange(0, 24, 1)
x_stop = np.arange(1, 21, 1)
x_dur = np.arange(1, 66, 1)


dist_lo = trapecia(x_dist, [0, 0, 7, 12])
dist_md = trapecia(x_dist, [8, 11, 19, 24])
dist_hi = trapecia(x_dist, [20, 22, 30, 30])

hour_nig = trapecia(x_hour, [0, 0, 6, 8])
hour_mor = trapecia(x_hour, [6, 9, 10, 12])
hour_day = trapecia(x_hour, [10, 12, 17, 19])
hour_eve = trapecia(x_hour, [17, 20, 23, 23])
# hour_mor_pic = triangle(x_hour, [7, 8, 9])
# hour_eve_pic = triangle(x_hour, [16, 17, 18])

stop_lo = trapecia(x_stop, [1, 1, 4, 6])
stop_md = triangle(x_stop, [4, 10, 16])
stop_hi = trapecia(x_stop, [14, 16, 20, 20])

dur_lo = trapecia(x_dur, [1, 1, 17, 30])
dur_md = triangle(x_dur, [20, 35, 50])
dur_hi = trapecia(x_dur, [40, 55, 65, 65])

fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2, figsize=(8, 9))

ax0.plot(x_dist, dist_lo, 'b', linewidth=1.5, label='Low')
ax0.plot(x_dist, dist_md, 'g', linewidth=1.5, label='Mid')
ax0.plot(x_dist, dist_hi, 'r', linewidth=1.5, label='High')
ax0.set_title('Distance')
ax0.legend()

ax1.plot(x_hour, hour_nig, 'b', linewidth=1.5, label='Night')
ax1.plot(x_hour, hour_mor, 'y', linewidth=1.5, label='Morning')
ax1.plot(x_hour, hour_day, 'c', linewidth=1.5, label='Day')
ax1.plot(x_hour, hour_eve, 'k', linewidth=1.5, label='Evening')
# ax1.plot(x_hour, hour_mor_pic, 'g', linewidth=1.5, label='v')
# ax1.plot(x_hour, hour_eve_pic, 'r', linewidth=1.5, label='vz')
ax1.set_title('Time of day')
ax1.legend()

ax2.plot(x_stop, stop_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_stop, stop_md, 'g', linewidth=1.5, label='Mid')
ax2.plot(x_stop, stop_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Number of stops')
ax2.legend()

ax3.plot(x_dur, dur_lo, 'b', linewidth=1.5, label='Low')
ax3.plot(x_dur, dur_md, 'g', linewidth=1.5, label='Mid')
ax3.plot(x_dur, dur_hi, 'r', linewidth=1.5, label='High')
ax3.set_title('Traveling duration')
ax3.legend()

plt.show()


dist_level_lo = ProintPosTrap(5, 0, 0, 7, 12)
dist_level_md = ProintPosTrap(5, 8, 11, 19, 24)
```

```
dist_level_hi = ProintPosTrap(5, 20, 22, 30, 30)

hour_level_nig = ProintPosTrap(6.5, 0, 0, 6, 8)
hour_level_mor = ProintPosTrap(6.5, 6, 9, 10, 12)
hour_level_day = ProintPosTrap(6.5, 10, 12, 17, 19)
hour_level_eve = ProintPosTrap(6.5, 17, 20, 23, 23)

stop_level_lo = ProintPosTrap(13, 1, 1, 4, 6)
stop_level_md = pointPosTri(13, 4, 10, 16)
stop_level_hi = ProintPosTrap(13, 14, 16, 20, 20)

active_rule1 = np.fmax(dist_level_hi, stop_level_hi)
active_rule2 = np.fmin(dist_level_lo, hour_level_nig)
active_rule3 = np.fmin(dist_level_md, stop_level_hi)
active_rule4 = np.fmin(stop_level_lo, dist_level_lo)
active_rule5 = np.fmax(dist_level_md, stop_level_md)
active_rule6 = np.fmax(hour_level_mor, stop_level_hi)
active_rule7 = np.fmin(dist_level_hi, hour_level_day)
active_rule8 = np.fmin(dist_level_md, stop_level_hi)
active_rule9 = np.fmax(stop_level_lo, hour_level_nig)
active_rule10 = np.fmin(dist_level_md, stop_level_lo)
active_rule11 = np.fmin(stop_level_md, dist_level_hi)
active_rule12 = np.fmin(hour_level_day, dist_level_lo)
active_rule13 = np.fmin(dist_level_lo, stop_level_md)

rule_low = max([active_rule2, active_rule4, active_rule9, active_rule12])

rule_md = max([active_rule5, active_rule7, active_rule10, active_rule13])

rule_hi = max([active_rule1, active_rule3, active_rule6, active_rule8,
active_rule11])

insurance_activation_lo = np.fmin(rule_low, dur_lo)


insurance_activation_md = np.fmin(rule_md, dur_md)

insurance_activation_hi = np.fmin(rule_hi, dur_hi)

insuranceZeros = np.zeros_like(x_dur)

fig, ax0 = plt.subplots(figsize=(8, 3))

ax0.fill_between(x_dur, insuranceZeros, insurance_activation_lo, facecolor='b',
alpha=0.7)
ax0.plot(x_dur, dur_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.fill_between(x_dur, insuranceZeros, insurance_activation_md, facecolor='b',
alpha=0.7)
ax0.plot(x_dur, dur_md, 'y', linewidth=0.5, linestyle='--')
ax0.fill_between(x_dur, insuranceZeros, insurance_activation_hi, facecolor='b',
alpha=0.7)
ax0.plot(x_dur, dur_hi, 'g', linewidth=0.5, linestyle='--')

plt.tight_layout()
plt.show()

aggregated = np.fmax(insurance_activation_lo, np.fmax(insurance_activation_md,
insurance_activation_hi)) # taisykliu sarasas
```

```
defuzz_centroid = centroid(x_dur, aggregated)
defuzz_mom = MOM(x_dur, aggregated)

print("Centroid metodas - siūloma alga programuotojui: " + str(defuzz_centroid))
print("Bisector metodas - siūloma alga programuotojui: " + str(defuzz_mom))
```