# Poglavje 10

# Naravno procesiranje

Področje *naravnega procesiranja* (angl. *natural computation*) razdelimo na naslednja tri različna polja raziskav:

- *procesiranje inspirirano z zakonitostmi iz naravnega okolja* (angl. *biologically inspired computing*): v tem primeru jemljemo ideje za algoritme iz narave; takšno procesiranje naj bi reševalo kompleksne probleme, ki jih do sedaj nismo znali reševati; tipični primeri povzemanja idej iz naravnega okolja za tvorbo algoritmov so *evolucijske metode* (angl. *evolutionary computation*), *metode genetskih algoritmov* (angl. *genetic algorithms*), *metode mehke logike* (angl. *fuzzy logic*) in *metode umetnih nevronskih mrež* (angl. *artificial neural neworks*); mednje sodita tudi že predhodno opisana koncepta *tropizma* in *feromonov* iz poglavja o amorfnem procesiranju;

- *modeliranje in simulacije procesov iz naravnega okolja*: v tem primeru želimo zgraditi algoritme, ki bi bili v največji možni meri zmožni simulirati procese iz naravnega okolja; tipični primeri tega področja so *fraktalska geometrija*, *skupinsko obnašanje* (angl. *collective behaviour*), *umetno življenje* (angl. *artificial life*) in novejše področje *kognitivnega procesiranja*[1] (angl. *cognitive computing*);

- *procesiranje v naravnih materialih* (angl. *hardware gnostic*): v tem primeru iščemo platforme iz naravnega okolja, na katerih bi bilo možno izvajati nadzorovano (programirano) procesiranje; del tega področja smo osvetlili v poglavjih o DNA in kvantnem procesiranju;

Vsa navedena področja so med seboj lahko tudi prepletena, tako da posamezna metoda lahko sodi v več področij. V pričujočem poglavju si bomo ogledali nekaj zgledov iz prve in druge skupine naštetih alinej.

[1] Kognitivno procesiranje se ukvarja s področjem modeliranja človeškega zaznavanja, sklepanja in odzivanja na stimulanse.

## 10.1 Fraktalsko procesiranje

Fraktalsko procesiranje sodi v delitvi področij naravnega procesiranja v kategorijo metod za modeliranje in simulacije procesov iz naravnega okolja. Na samem začetku navedimo neposredni izsek teksta iz vira [100], ki nam bo služil kot osnovna motivacija za pristop k fraktalskemu procesiranju.

> *Vzemimo, da sedimo ob Jadranski obali in opazujemo, kako se morje dotika obale. Ker imamo ravno čas, razmišljamo, kako dolga je pravzaprav obala med Trstom in Dubrovnikom. Vzemimo merilo dolžine $r$ in ga polagajmo ob prelepi obali z namenom, da zmerimo resnično dolžino tako, da na koncu merila sledi začetek merila. Dobimo izmerjeno dolžino obale $L(r) = Nr$, če je $N$ število polaganj merila na obalo. Dobljena dolžina $L(r)$ ni enaka resnični dolžini obale, saj je ta slučajno razgibana (nagubana in zakrivljena), merilo, s katerim merimo, pa je ravno in se ujema z resnično obalo samo v nekaj ali samo v dveh točkah. Vemo le to, da je resnična dolžina obale večja od $L(r)$, za koliko, seveda ne vemo. Če merilo $r$ zmanjšujemo, se nam pri merjenju $N$ povečuje, do prave dolžine obale pa s takšnim merjenjem ne pridemo. Človek bi pričakoval, da se po velikem številu korakov manjšanja merila izmerjena dolžina $L(r)$ približa resnični dolžini obale. Pa temu ni tako. Izkaže se, da se izmerjena dolžina sicer povečuje vendar pa ne limitira v limito, ki bi ji lahko rekli resnična dolžina obale. Da je temu tako, si lahko razlagamo s povečevanjem slike delčka obale na normalno velikost slike obale. Pri normalni velikosti slike delčka obale, imamo podobno razgibanost obale, kot je na sliki celotne obale. V fraktalnost nas vodi prav omenjena podobnost razgibanosti in omenjena nelimitnost.*

Problem opisan v citiranem izseku teksta že v šestdesetih letih prejšnjega stoletja izpostavi pionir fraktalskega procesiranja Benoît Mandelbrot. Ob predpostavki, da je dolžina obale $L(r)$ pogojena z dolžino merila $r$ in številom polaganj merila $N$ ($L(r) = r * N$), pridemo do izraza

$$\text{if } (r_{i+1} < r_i) \text{ then } (L(r_{i+1}) \geq L(r_i)), \tag{10.1}$$

pri čemer $r_{i+1}$ predstavlja dolžino merila, ki ga uporabljamo ob izvajanju $i+1$ ponovitve meritve. Z zaporedjem izvajanja meritev, pri čemer bi bilo merilo vse krajše, bi prišli do izraza

$$L(r_1) \leq L(r_2) \leq ... \leq N(r_i) \leq ... \tag{10.2}$$

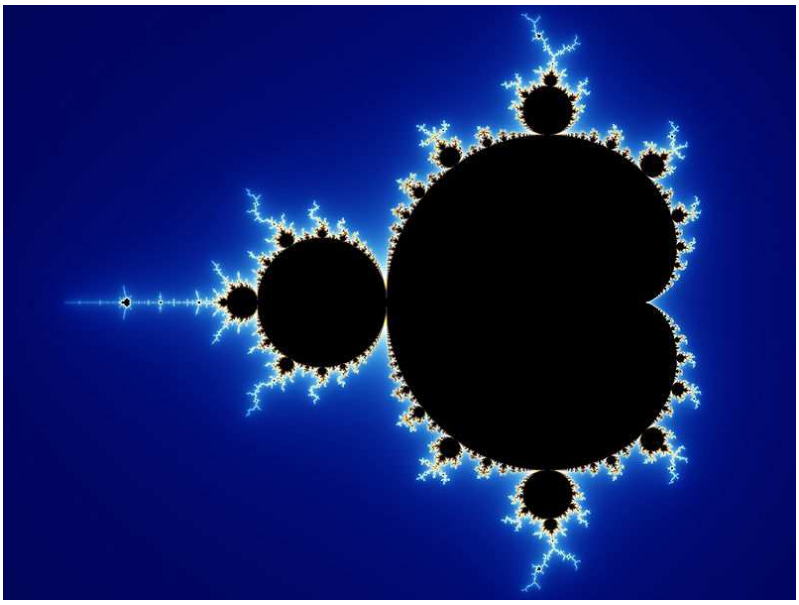
Očitno je kompleksnost narave tako velika, da nam je odgovor na vprašanje o dolžini obale nedosegljiv. Navkljub temu smo ob predstavitvi problema naleteli na pojma *razgibanosti* in *podobnosti* posameznih oblik obale, ki jih srečujemo v naravnem okolju neodvisno od nivoja opazovanja. Prav na značilnostih podobnosti in razgibanosti temelji področje fraktalskega procesiranja, čigar osnova je

*fraktalska geometrija*, rezultati fraktalskega procesiranja pa so *fraktalske strukture* ali enostavno *fraktali*. Definicijo fraktalske strukture navedemo v nadaljevanju.

**Definicija 11** *Fraktalska struktura ali fraktal je fragmentirana geometrična oblika, ki jo lahko razdelimo na dele. Vsak od delov je vsaj do neke mere reducirana kopija celotne strtukture.*

Iz definicije, ki je sicer „mehka“ je razvidno, da fraktalske strukture temeljijo na značilnosti *samopodobnosti*[2] (angl. *self similarity*).

Procesiranje fraktalskih struktur temelji na fraktalski geometriji, slednja pa v večini primerov na iterativnih matematičnih funkcijah (angl. *iterated function system* - IFS). Kot vzorčni primer fraktalske strukture je na sliki 10.1 predstavljena struktura Mandelbrotove množice.



Slika 10.1: Fraktalska struktura Mandelbrotove množice.

Karakteristične značilnosti fraktalskih struktur so sledeče:

- končna fraktalska struktura je poljubne velikosti;

[2]Sámopodóbnost v matematiki ponazarja objekte, ki so strogo ali približno podobni delu samega sebe, kar pomeni, da ima celota enako obliko kot en ali več njenih delov (Vir: Wikipedia).

- fraktalska struktura je praviloma neregularna, iz česar sledi, da je težko opisljiva v evklidski geometriji;
- fraktalska struktura ima vsaj stohastične lastnosti samopodobnosti (angl. *self similar*);
- fraktalska struktura je definirana z rekurzivno ali iterativno definicijo;

V fraktalskih strukturah poznamo tri osnovne vrste samopodobnosti. Le te so sledeče:

- *eksaktna samopodobnost* (angl. *exact self similarity*): gre za najstrožjo vrsto sampodobnosti, pri katerih morajo biti posamezni deli strukture enaki na različnih velikostnih nivojih; običajno so tovrstne strukture pridobljene na osnovi iterativnih funkcij;
- *delna samopodobnost* (angl. *quasi self similarity*): samopodobnost na različnih velikostnih nivojih je le približna; običajno so tovrstne strukture pridobljene na osnovi rekurzivnih funkcij;
- *statistična samopodobnost* (angl. *statistical self similarity*): podobnost struktur posameznih velikostnih razredov je pogojena statistično;

Tipični primeri naravnih pojavov, ki so vizualno izrazljivi z fraktalskimi strukturami so oblike oblakov, gora, obal, snežink, listov dreves itd. V nadaljevanju predstavimo tri vzorčne primere enostavne tvorbe fraktalskih struktur.

### 10.1.1 Kochova snežinka

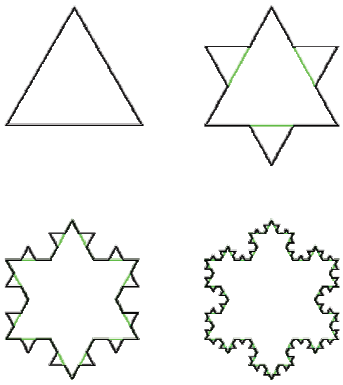
Fraktalska struktura Kochove snežinke je ena od najstarejših fraktalskih struktur, ki jo leta 1904 predstavi njen avtor N.F. von Koch. Psevdo algoritem za njeno tvorbo je predstavljen v izpisu 10.1, grafična ponazoritev njene evolucije pa na sliki 10.2.

```
1 - začetno strukturo ponazori z enakostraničnim trikotnikom;
2 - ponavljaj za vse trikotnike:
3   - vsako od stranic trikotnika razdeli na tri enako dolge daljice;
4   - srednjo daljico zbriši;
5   - na mesto izbrisane daljice postavi nov enakostranični trikotnik
        po naslednjih pravilih:
6     - njegova stranica naj bo po dolžini enaka dolžini izbrisane
      daljice;
7     - trikotnik naj bo usmerjen iz obstoječe strukture;
8     - nosilna stranica trikotnika naj bo nevidna;
```

Listing 10.1: Psevdo algoritem za tvorbo fraktalske strukture Kochove snežinke.

### 10.1.2 Trikotniki Sierpinskega

Fraktalsko strukturo trikotnikov Sierpinskega predstavi l. 1915 njen avtor W. Sierpinski. Psevdo algoritem za njihovo tvorbo je predstavljen v izpisu 10.2, grafična ponazoritev njihove evolucije pa na sliki 10.3.



Slika 10.2: Evolucija fraktalske strukture Kochove snežinke (vir slike: Wikipedia).

```
– začetno strukturo ponazori s črnim enakostraničnim trikotnikom;
– ponavljaj za vsak črn trikotnik:
  – v črn trikotnik postavi enakostraničen bel trikotnik, čigar
    stranica je za polovico manjša od stranice črnega;
```

Listing 10.2: Psevdo algoritem za tvorbo fraktalske strukture trikotnikov Sierpinskega.

Slika 10.3: Evolucija fraktalske strukture trikotnikov Sierpinskega (vir slike: Wikipedia).

### 10.1.3 Generator fraktalske strukture praproti

Fraktalsko strukturo praproti predstavi M. Barnsley l. 1993 (angl. *Barnsley's fern*). Osnova njenega generatorja je nabor iterativnih funkcij, predstavljenih z izrazi

$$x_{n+1} = 0, y_{n+1} = 0,16 * y_n, \tag{10.3}$$

$$x_{n+1} = 0,2 * x_n - 0,26 * y_n, y_{n+1} = 0,23 * x_n + 0,22 * y_n + 1,6, \tag{10.4}$$

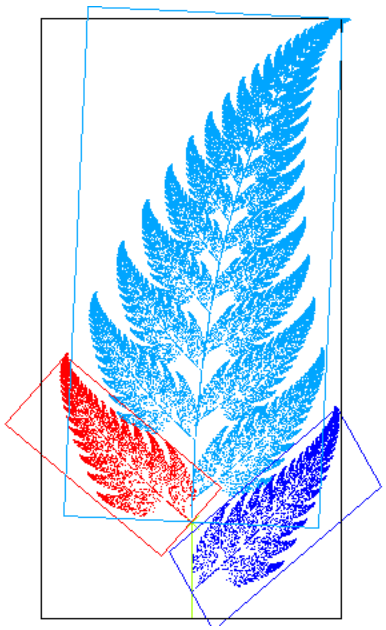
$$x_{n+1} = -0,15 * x_n + 0,28 * y_n, y_{n+1} = 0,26 * x_n + 0,24 * y_n + 0,44, \quad (10.5)$$

$$x_{n+1} = 0,85 * x_n + 0,04 * y_n, y_{n+1} = -0,04 * x_n + 0,85 * y_n + 1,6, \quad (10.6)$$

pri čemer velja tudi izraz

$$x_0 = y_0 = 0, \quad (10.7)$$

ki inicializira koordinati prve točke fraktalske strukture v dvodimenzionalnem prostoru. Vse nadaljne koordinate točke $(x_{n+1}, y_{n+1})$ izračunamo po enem od predhodno navedenih izrazov, pri čemer je izbira posameznega izraza za izračun nove koordinate točke pogojena verjetnostno. Prvi izraz (10.3) je izbran v 1% iteracij, drugi ali tretji izraz ((10.4) ali (10.5)) v 7% iteracij, zadnji četrti izraz (10.6) pa v 85% iteracij. Končni rezultat generiranja fraktalske strukture praproti je predstavljen na sliki 10.4. Pri tem izraz (10.3) zagotavlja forma-



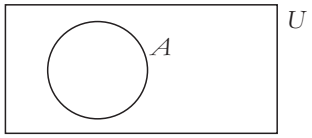
Slika 10.4: Fraktalska struktura lista praproti.

cijo podstrukture stebla praproti zelene barve, izraz (10.4) formacijo podstrukture rdečega lista, izraz (10.5) formacijo podstrukture temnomodrega lista, izraz (10.6) pa formacijo osrednjega svetlomodrega lista. Pri tem je koordinatni sistem okvira na sliki 10.4 omejen na prostor koordinat, predstavljen z izrazom

$$x_i \in [-5, 5], y_i \in [0, 10], i \in \mathbb{N}_0. \quad (10.8)$$

## 10.2 Mehka logika

Predpostavimo, da imamo opravka z univerzalno množico elementov $U$, v okviru katere imamo definirano množico elementov $A$, kar je grafično ponazorjeno na sliki 10.5. Klasična teorija množic trdi, da nek element $x$, ki pripada univerzalni




Slika 10.5: Grafična ponazoritev univerzalne množice elementov $U$ in v njenem okviru definirane množice $A$.

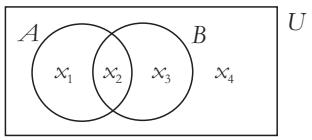
množici $U$ ($x \in U$), bodisi pripada množici $A$ ($x \in A$), bodisi pa tej množici ne pripada ($x \notin A$). Pri tem pod pojmom pripadnosti smatramo popolno pripadanje elementa k množici. Omenjena dejstva lahko zapišemo z izrazoma

$$x \in A : \mu_A(x) = 1, \tag{10.9}$$

$$x \notin A : \mu_A(x) = 0, \tag{10.10}$$

pri čemer izraz $\mu_A(x)$ predstavlja *stopnjo pripadnosti* elementa $x$ k množici $A$, izraz $\mu_A$ pa poimenujemo za *pripadnostno funkcijo* množice $A$. Slednja nam za opazovani element $x$ vrne njegovo stopnjo pripadnostni k množici $A$.

Upoštevajoč predstavljeno notacijo in zgled grafične ponazoritve pripadnosti elementov $x_1$, $x_2$, $x_3$ in $x_4$ k množicama $A$ in $B$ s slike 10.6 lahko zapišemo




Slika 10.6: Zgled grafične ponazoritve pripadnosti elementov $x_1$, $x_2$, $x_3$ in $x_4$ k množicama $A$ in $B$.

sledeče izraze:

$$x_1 \in A, x_1 \notin B : \mu_A(x_1) = 1, \mu_B(x_1) = 0, \tag{10.11}$$

$$x_2 \in A, x_2 \in B : \mu_A(x_2) = 1, \mu_B(x_2) = 1, \tag{10.12}$$

$$x_3 \notin A, x_3 \in B : \mu_A(x_3) = 0, \mu_B(x_3) = 1, \tag{10.13}$$

$$x_4 \notin A, x_4 \notin B : \mu_A(x_4) = 0, \mu_B(x_4) = 0. \tag{10.14}$$

Iz predstavljenega zgleda je razvidno, da zaloga vrednosti posamezne pripadnostne funkcije $\mu_A$ v **klasični teoriji množic** izhaja iz množice $\{0, 1\}$

($\mu_A \in \{0, 1\}$). Vrednosti 0 in 1 si lahko interpretiramo tudi kot vrednosti False ali True, ki opredeljujeta izjavo, da nek element $x$ pripada neki množici $A$.

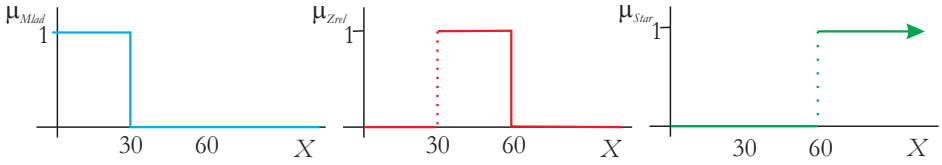
Na tem mestu si oglejmo primer klasifikacije skupine ljudi v množice, glede na njihovo starost. Predpostavimo, da ljudi klasificiramo v množice *Mlad*, *Zrel* in *Star* ter da so si množice paroma med seboj tuje[3]. Ob predpostavki, da spremenljivka $x$ predstavlja starost posameznega človeka v letih, bomo za potrebe klasifikacije uporabili sledeče izraze:

$$0 \leq x \leq 30 : x \in Mlad \tag{10.15}$$

$$30 < x \leq 60 : x \in Zrel \tag{10.16}$$

$$x > 60 : x \in Star \tag{10.17}$$

Na osnovi podanih izrazov lahko grafično ponazorimo pripadnostne funkcije za vse tri množice, kar predstavimo na sliki 10.7.




Slika 10.7: Grafična ponazoritev pripadnostnih funkcij $\mu_{Mlad}$, $\mu_{Zrel}$ in $\mu_{Star}$.

Predpostavimo, da imamo opravka s tremi ljudmi starimi 1 leto ($x_1 = 1$), 29 let ($x_2 = 29$) in 31 let ($x_3 = 31$). Glede na postavljene klasifikacijske kriterije lahko ugotovimo, da prvi dve osebi med katerima je razlika kar 28 let sodita v isto množico mladih ljudi ($\mu_{Mlad}(1) = \mu_{Mlad}(29) = 1$), zadnja oseba pa sodi v množico zrelih ljudi ($\mu_{Zrel}(31) = 1$), čeprav je starostna razlika med drugo in tretjo osebo zgolj dve leti in sta si v starostnem smislu osebi dve in tri dosti bolj podobni, kot osebi ena in dve. Takšna klasifikacija podatkov v množice je ostra (angl. *crisp*) in vodi v ostro logiko (angl. *crisp logic*), pri čemer množice, ki temeljijo na pripadnostnih funkcijah iz izraza

$$\mu_A(x) \in \{0, 1\} \tag{10.18}$$

imenujemo za *ostre množice* (angl. *crisp set*). Pri njihovi uporabi lahko prihaja do anomalij, kakršno smo predstavili na primeru klasifikacije starosti treh ljudi.

### 10.2.1 Mehke množice

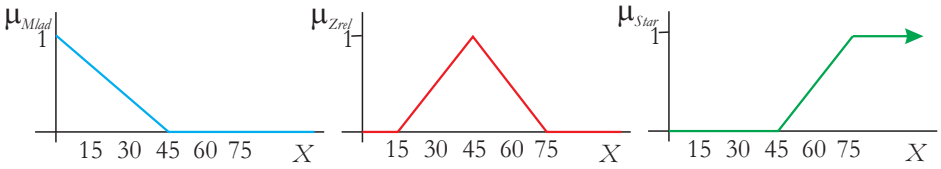
Teorijo *mehkih množic* (angl. *fuzzy set*) v šestdesetih letih prejšnjega stoletja postavi Lofti A. Zadeh. Osnovna ideja mehkih množic je, da dopuščajo tudi

[3]Množici $A$ in $B$ sta si med seboj tuji ali disjunktni, ko je njun presek prazen ($A \cap B = \{\}$).

nepopolne ali delne pripadnosti opzovanih elementov univerzalne množice k neki opazovani mehki množici. V tem primeru velja izraz

$$\mu_A(x) \in [0,1]\,, \tag{10.19}$$

s čimer postane zaloga vrednosti pripadnostne funkcije k mehki množici zvezni interval od 0 do 1. Na tem mestu lahko za zgled s slike 10.7 definiramo drugačen nabor pripadnostnih funkcij, ki bodo zavzemale tudi neke vmesne vrednosti in s tem posredno definirale neko *delno* oziroma *nepopolno pripadnost* opazovanega elementa k opazovani mehki množici. Primer drugače definiranih oziroma mehkih pripadnostnih funkcij je predstavljen na sliki 10.8. V podanem zgledu smo




Slika 10.8: Zgled postavitve pripadnostnih funkcij $\mu_{Mlad}$, $\mu_{Zrel}$ in $\mu_{Star}$ v domeni mehke logike.

za definicije pripadnostnih funkcij uporabili odsekoma konstantne ter odsekoma linearne funkcije, pri čemer bi lahko izbrali tudi običajne zvezne funkcije.

Z uporabo mehkih množic lahko preidemo na domeno *mehke logike* (angl. *fuzzy logic*), zgled uporabe slednje pa si ogledamo v naslednjem razdelku. Uporaba mehke logike nam omogoča izvajanje *decizije*[4] na osnovi nepreciznih, nejasnih, nenumeričnih, lingvistično opisanih podatkov.

### 10.2.2 Zgled mehkega krmilnika temperature

V pričujočem razdelku bomo predstavili zgled mehkega krmilnika temperature, kot tipičen primer uporabe mehke logike pri izvajanju mehkega odločanja (angl. *fuzzy decision*). Za cilj si zadajmo logično realizacijo krmilnika, ki bo na osnovi krmiljenja grelca vzdrževal željeno ambientalno temperaturo 20°C v opazovanem prostoru. Krmilnik bo deloval v diskretnih časovnih korakih, pri čemer naj bi na $k$-tem koraku na vhodu prejel vrednost ambientalne temperature $T_k$ ter na osnovi nje na izhod oddal spremembo moči delovanja grelca $\Delta Out_k$. Predpostavimo, da grelec lahko krmilimo v zveznem razponu moči njegovega delovanja od 0% do 100% moči njegovega delovanja. Tako lahko zapišemo izraz

$$Out_k = Out_{k-1} + \Delta Out_k, Out_k \in [0,100]\,, \tag{10.20}$$

s katerim ponazorimo, da bo izhodna moč gretja na $k$-tem koraku oziroma $k$-tem časovnem intervalu $Out_k$ spremenjena za faktor $\Delta Out_k$, glede na izračunano

[4] Decizija (angl. *decision*) - odločitev, odločanje.

moč gretja $Out_{k-1}$ iz predhodnega časovnega intervala izdelave decizije. Poleg vhodne spremenljivke $T_k$ vpeljemo še drugo vhodno spremenljivko $\Delta T_k$, ki se izračuna po izrazu

$$\Delta T_k = T_k - T_{k-1}, \tag{10.21}$$

ter odraža tendence rasti, vzdrževanja ali padanja temperature glede na zadnji dve zajeti vrednosti ambientalne temperature iz okolja. Vrednost vhodne spremenljivke $\Delta T_k$ se lahko izračuna v samem krmilniku, če le ta ima zmožnost pomnjenja predhodne zajete vhodne vrednosti. Shema opisanega krmilnika je predstavljena na sliki 10.9, pri čemer modul $M$ izračuna izraz (10.21) in pomni predhodno zajeto temperaturo $T_{k-1}$, modul $O$ pa izvede izračun izraza (10.20) in pomni predhodno izhodno vrednost $Out_{k-1}$.

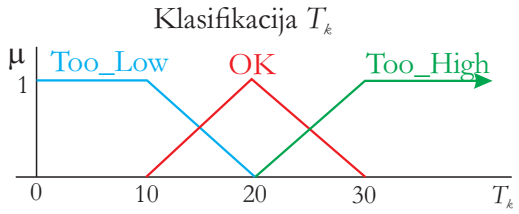

Slika 10.9: Shema krmilnika za uravnavanje željene ambientalne temperature.

V naslednjih razdelkih opišemo procese mehčanja (angl. *fuzzification*), izdelave mehke decizije (angl. *fuzzy inference*) in ostrenja rezultatov decizije (angl. *defuzzification*), da bi prišli do končne absolutne vrednosti izhodne moči grelca na $k$-tem koraku.

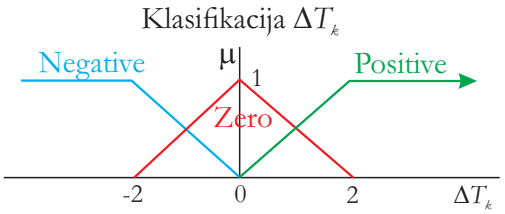
### Proces mehčanja

Mehki krmilnik mora v procesu mehčanja iz ostrih oziroma numeričnih vrednosti vhodnih spremenljivk $T_k$ in $\Delta T_k$ izračunati stopnje pripadnosti vnaprej definiranim mehkim množicam. V ta namen moramo najprej definirati pripadnostne funkcije mehkih množic za vsako opazovano spremenljivko. Pri tem se intuitivno odločimo za naslednje nabore mehkih množic za posamezno opazovano spremenljivko:

- vhodna spremenljivka absolutne ambientalne temperature $T_k$ bo definirana z mehkimi množicami $\{Too\_Low, OK, Too\_High\}$, njihove pripadnostne funkcije pa so definirane grafično na sliki 10.10;
- vhodna spremenljivka tendence gibanja ambientalne temperature $\Delta T_k$ bo definirana z mehkimi množicami $\{Negative, Zero, Positive\}$, njihove pripadnostne funkcije pa so definirane grafično na sliki 10.11;
- izhodna spremenljivka spremembe moči ogrevanja $\Delta Out_k$ bo definirana z mehkimi množicami $\{Negative, Zero, Positive\}$, njihove pripadnostne funkcije pa so definirane grafično na sliki 10.12; pri tem sta pripadnostni funkciji $Negative$ in $Positive$ omejeni na obeh straneh;




Slika 10.10: Definicije pripadnostnih funkcij za vhodno spremenljivko $T_k$.




Slika 10.11: Definicije pripadnostnih funkcij za vhodno spremenljivko $\Delta T_k$.

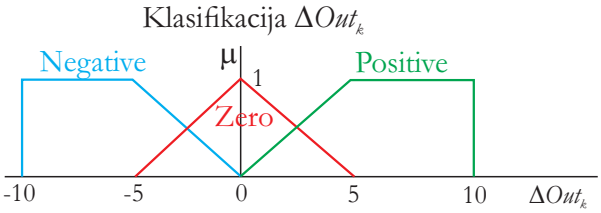
Predpostavimo, da na $k$-tem koraku krmilnik zajame vrednost ambientalne temperature 15°C ($T_k = 15$), predhodna zajeta temperatura pa je bila 13,5°C ($T_{k-1} = 13,5$, $\Delta T_k = 1,5$). V tem primeru lahko za vrednost spremenljivke $T_k$ zapišemo izraz

$$\mu_{Too\_Low}(15) = 0,5, \mu_{OK}(15) = 0,5, \mu_{Too\_High}(15) = 0, \tag{10.22}$$

za vrednost spremenljivke $\Delta T_k$ pa izraz

$$\mu_{Negative}(1,5) = 0, \mu_{Zero}(1,5) = 0,25, \mu_{Positive}(1,5) = 0,75, \tag{10.23}$$

s katerima ponazorimo pripadnosti konkretnih vrednosti spremenljivk $T_k$ ter $\Delta T_k$ k njunim mehkim množicam.




Slika 10.12: Definicije pripadnostnih funkcij za vhodno spremenljivko $\Delta Out_k$.

### Proces izdelave decizije

Proces izdelave decizije oziroma mehke inference (angl. *fuzzy decision*, *fuzzy inference*) v mehki logiki običajno temelji na uporabi lingvističnih pravil, katerih parametri izvirajo iz imen predhodno definiranih mehkih množic. Za krmiljenje ambientalne temperature bomo uporabili intuitivno določena lingvistična pravila predstavljena v izpisu 10.3.

```
1 if (T_k = Too_Low)  AND (ΔT_k = Negative)  then (ΔOut_k = Positive);
2 if (T_k = Too_Low)  AND (ΔT_k = Zero)      then (ΔOut_k = Positive);
3 if (T_k = Too_Low)  AND (ΔT_k = Positive)  then (ΔOut_k = Zero);
4 if (T_k = OK)       AND (ΔT_k = Negative)  then (ΔOut_k = Zero);
5 if (T_k = OK)       AND (ΔT_k = Zero)      then (ΔOut_k = Zero);
6 if (T_k = OK)       AND (ΔT_k = Positive)  then (ΔOut_k = Zero);
7 if (T_k = Too_High) AND (ΔT_k = Negative)  then (ΔOut_k = Zero);
8 if (T_k = Too_High) AND (ΔT_k = Zero)      then (ΔOut_k = Negative);
9 if (T_k = Too_High) AND (ΔT_k = Positive)  then (ΔOut_k = Negative);
```

Listing 10.3: Lingvistična pravila krmilnika za krmiljenje ambientalne temperature.

Predstavljena mehka pravila so sestavljena iz levega *pogojnega dela* in iz desnega *sklepnega dela*. Pogojni del je v našem primeru sestavljen iz preverjanja veljavnosti dveh pogojev - preverjanja velikosti ambientalne temperature $T_k$ in preverjanja tendence gibanja temperature $\Delta T_k$. Samo preverjanje veljavnosti posameznega pogoja se v obeh primerih izvaja po principu izračuna stopnje pripadnosti vstopajoče ostre vrednosti spremenljivke k opazovani mehki množici, ki nastopa v posameznem pogoju. Sestavljeni pogojni del pravila je sestavljen iz dveh pogojev, ki smo ju povezali z operatorjem AND, kar pomeni, da je sklepni del pravila veljaven le, če veljata oba pogoja[5].

Na tem mestu vpeljemo pojem *relevantnosti* ali *doprinosa* posameznega mehkega pravila k končni deciziji glede na izpolnjenosti pogojev iz pogojnega dela pravila. Relevantnost posameznega pravila enačimo z *minimumom* pripadnosti ostrih vrednosti k mehkim množicam iz posameznih pogojev opazovanega pravila[6].

Predpostavimo, da smo pred izdelavo decizije na $k$-tem koraku, pri čemer je vstopajoča ostra vrednost na tem koraku $T_k = 15$, na predhodnem koraku pa je bila $T_{k-1} = 13,5$. Za začetek vzemimo pod drobnogled pravilo št.2 iz izpisa 10.3. Prvi pogoj v pravilu doprinese k relevantnosti tega pravila vrednost 0,5 ($\mu_{Too\_Low}(15) = 0,5$), drugi pogoj v tem pravilu pa relevantnost 0,25 ($\mu_{Zero}(1,5) = 0,25$). Ker smo za sestavo pogojnega dela pravila uporabili operator AND, v tem primeru izračunamo minimum obeh vrednosti, s čimer smo dobili relevantnost celotnega pravila št.2 na $k$-tem koraku decizije. Slednjo izračunamo po izrazu

$$min(\mu_{Too\_Low}(15), \mu_{Zero}(1,5)) = min(0,5,\ 0,25) = 0,25. \tag{10.24}$$

[5] Poleg operatorja AND v mehkih pravilih pogosto uporabljamo tudi operatorja OR in NEG.

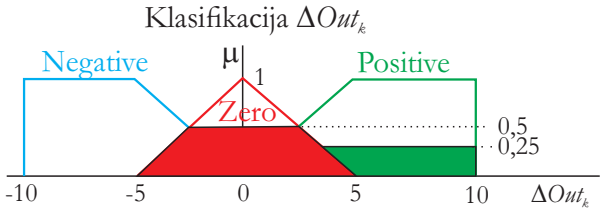
[6] V primeru, da bi uporabili vezni operator OR, bi relevantnost opazovanega pravila izračunali kot maksimum pripadnosti ostrih vrednosti k mehkim množicam iz posameznih pogojev opazovanega pravila.

| Št.pravila | $\mu_{Left}$ | $\mu_{Right}$ | $min(\mu_{Left}, \mu_{Right})$ | Sklep |
|---|---|---|---|---|
| 1 | $0,5$ | 0 | 0 | *Positive* |
| 2 | $0,5$ | $0,25$ | $0,25$ | *Positive* |
| 3 | $0,5$ | $0,75$ | $0,5$ | *Zero* |
| 4 | $0,5$ | 0 | 0 | *Zero* |
| 5 | $0,5$ | $0,25$ | $0,25$ | *Zero* |
| 6 | $0,5$ | $0,75$ | $0,5$ | *Zero* |
| 7 | 0 | 0 | 0 | *Zero* |
| 8 | 0 | $0,25$ | 0 | *Negative* |
| 9 | 0 | $0,75$ | 0 | *Negative* |

Tabela 10.1: Tabela rezultatov evaluacije sklepov posameznih pravil z njihovimi relevancami.

Ob izdelave decizije na $k$-tem koraku moramo seveda evaluirati relevantnosti vseh pravil. V tabeli 10.1 so predstavljeni rezultati evaluacije relevantnosti pravil, pri čemer $\mu_{Left}$ predstavlja doprinos prvega pogoja v pravilu, $\mu_{Right}$ doprinos drugega pogoja v pravilu, $min(\mu_{Left}, \mu_{Right})$ pa celotno relevantnost pravila.

Glede na vrednosti iz tabele 10.1 lahko ugotovimo, da so na $k$-tem koraku *relevantna* le pravila 2, 3, 5 in 6[7], pri čemer prvo od navedenih pravil ponuja za sklep lingvistični termin *Positive*, preostali trije pa mehki sklep *Zero*. Končni mehki sklep navedene relevatne množice pravil na $k$-tem koraku predstavimo na sliki 10.13.




Slika 10.13: Mehki sklep relevantne množice pravil na $k$-tem koraku.

Mehki sklep je predstavljen z obarvanim delom mehkih množic *Positive* in *Zero*. Pri tem je mehka množica *Positive* obarvana samo do njene četrtine (relevantnost pravila št.2 je 0,25), mehka množica *Zero* pa do polovice, saj pravila 3, 5 in 6 doprinesejo relevantnosti 0,5, 0,25 ter 0,5, mi pa za barvanje uporabimo največjo relevanco teh treh pravil in sicer 0,5. S tem smo prišli do dokončnega mehkega sklepa o mehki vrednosti spremenljivke $\Delta Out_k$, ki ga predstavljata oba pobarvana lika.

[7] Za relevantna pravila štejemo zgolj tista, pri katerih je vrednost $min(\mu_{Left}, \mu_{Right})$ večja od nič.

**Proces ostrenja odločitve**

Primarni cilj mehkega krmilnika je določitev ostre vrednosti spremenljivke $\Delta Out_k$, na osnovi katere bomo lahko na $k$-tem koraku evaluirali izraz

$$Out_k = Out_{k-1} + \Delta Out_k. \tag{10.25}$$

V ta namen moramo mehki sklep, grafično ponazorjen na sliki 10.13 v obliki ploščine sestavljene iz dveh likov, transformirati v ostro vrednost. Tovrsti postopek imenujemo za *ostrenje* (angl. *defuzzification*). V večini primerov za ostrenje uporabljamo metodo *izračuna težišča sestavljenega lika* (angl. *center of gravity* - COG). Ker smo pripadnostne funkcije definirali kot kombinacijo odsekoma linearnih in odsekoma konstantnih funkcij, lahko težišče sestavljenega lika izračunamo na osnovi izraza

$$\Delta Out_k = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} y_i}, \tag{10.26}$$

ki temelji na končnih vsotah. Pri tem $x_i$ predstavlja $i$-to vrednost na $x$ osi na grafu s slike 10.13, $y_i$ pa višina obarvanega segmenta pri vrednosti $x_i$. Ob prepostavki, da $i$ teče po celih številih, bi tako v našem primeru prišli do vrednosti parametrov v tabeli 10.2, končni rezultat ostrenja pa podamo z izrazom

$$\Delta Out_k = \frac{11,45}{5,25} = 2,18. \tag{10.27}$$

Za normalno delovanje krmilnika bi morali pred samo aktivacijo krmilnika določiti še začetno vrednost moči gretja $Out_0$. Mehki krmilniki so zamenjali klasične PID[8] krmilnike v mnogih aplikacijah. Odlikujeta jih enostavnost postavitve in človeškemu razmišljanju podobna logika.

Več informacij o osnovah mehke logike bralec najde v viru [101]. Po kategorizaciji metod v različne skupine naravnega procesiranja bi mehka logika kot metoda sodila v prvo klasifikacijsko skupino.

## 10.3 Modeliranje požara v naravnem okolju na osnovi mehke logike

V pričujočem razdelku predstavimo model širjenja požara v naravnem okolju na osnovi mehke logike. Njegov osnovni namen je na osnovi simulacije določiti obseg področja, ki ga bo požar zavzel po nekem pretečenem času od njegovega začetka.

Problemi požarov v naravnem okolju pestijo predvsem velike in redko poseljene države kot so npr. Kanada, Brazilija, Rusija itd. Požarne službe v teh

[8] PID krmilnik - proporcionalno integrirno diferencialni krmilnik.

| $x_i$ | $y_i$ | $x_i * y_i$ |
|---|---|---|
| $-5$ | $0,00$ | $0,00$ |
| $-4$ | $0,20$ | $-0,80$ |
| $-3$ | $0,40$ | $-1,20$ |
| $-2$ | $0,50$ | $-1,00$ |
| $-1$ | $0,50$ | $-0,50$ |
| $0$ | $0,50$ | $0,00$ |
| $1$ | $0,50$ | $0,50$ |
| $2$ | $0,50$ | $1,00$ |
| $3$ | $0,40$ | $1,20$ |
| $4$ | $0,25$ | $1,00$ |
| $5$ | $0,25$ | $1,25$ |
| $6$ | $0,25$ | $1,50$ |
| $7$ | $0,25$ | $1,75$ |
| $8$ | $0,25$ | $2.00$ |
| $9$ | $0,25$ | $2,25$ |
| $10$ | $0,25$ | $2,50$ |
| - | $\sum_{i=1}^{n} y_i = 5,25$ | $\sum_{i=1}^{n} x_i y_i = 11,45$ |

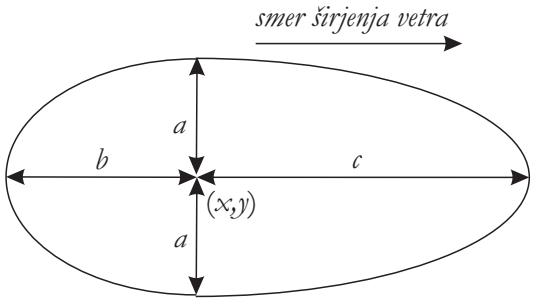
Tabela 10.2: Tabela vrednosti parametrov za izračun COG na $k$-tem koraku delovanja mehkega krmilnika.

državah ob izbruhu požara na osnovi modelov najprej izvedejo simulacije do kod in kako hitro se bo požar razširil, v nadaljevanju ocenijo materialno škodo požara in šele po teh dveh fazah pride do odločitve, ali je posredovanje ob požaru sploh smiselno oziroma ekonomsko opravičeno. Omenjene simulacije služijo tudi pri načrtovanju izdelave požarnih posekov, preko katerih se požar ne more širiti in pri načrtovanju eventuelnih evakuacij ljudi in njihove imovine pred posledicami požara.

Modele za širjenje požarov v naravnem okolju lahko razdelimo na tri osnovne skupine. Le te so sledeče:

- *fizikalno - kemijski modeli*: omenjeni modeli temeljijo na eksaktnih fizikalno - kemijskih enačbah in so najbolj natančni; za modeliranje širjenja na večjih površinah so praktično neuporabni, zaradi nerazpoložljivosti vseh vhodnih parametrov (npr. vrst vegetacije, vlažnosti vegetacije, vlažnosti ozračja, rosiščne temperature, ambientalne temperature itd.);

- *statistični modeli*: omenjeni modeli temeljijo na meritvah širjenja v umetno ustvarjenih okoljih (npr. vetrovnih tunelih) ali na meritvah širjenja realnih požarov; prvi je začel tovrstne meritve izvajati v prejšnjem stoletju R.C Rothermel, ki vzpostavi osnovno metodo predikcije, temelječo na dveh polelipsah, predstavljenih na sliki 10.14; ključni parametri, ki jih model napove glede na začetno točko požara v koordinati $(x, y)$, so




Slika 10.14: Rothermelova metoda predikcije na osnovi dveh polelips.

$a$, ki predstavlja razdaljo širjenja požara pravokotno na smer vetra, $b$, ki predstavlja razdaljo širjenja požara proti smeri vetra in $c$, ki predstavlja razdaljo širjenja požara v smeri vetra;

- *celični modeli*: celični modeli temeljijo na regularnem dvodimenzionalnem prostoru celic - avtomatov; njihova osnova temelji na celularnih avtomatih, pri čemer je *program* posameznega avtomata za prehajanji med njegovimi stanji kompleksnejši, kot smo jih bili vajeni v poglavju o celularnih avtomatih;

V pričujočem razdelku bomo za potrebe modeliranja požara v naravnem okolju predstavili model mehkega programa posameznega avtomata oziroma celice v dvodimenzionalni celični strukturi povzet po viru [102]. Model je bil razvit na FRI pred približno petnajstimi leti za potrebe pridruževanja Slovenije k organizaciji NATO.

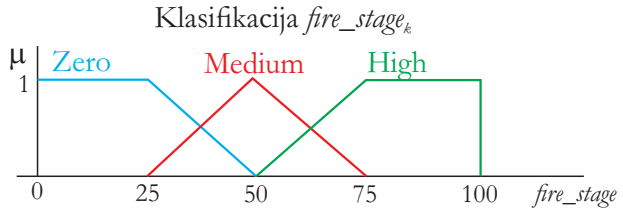
Predpostavimo, da imamo regularen dvodimenzionalni prostor celic, ki jih lahko naslavljamo preko njihovih koordinat $(x, y)$. Na tem mestu vpeljemo sestavljeno podatkovno strukturo cells, ki jo predstavimo v izpisu 10.4.

```
1 struct cells
2 { int fire_stage;       /* q(x,t) */
3   int inflammability;   /* q(x,t) */
4   int next_fire_stage;   /* q(x,t+1) */
5 } cell[max_x,max_y]
```

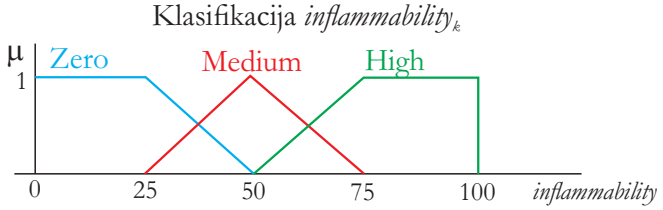
Listing 10.4: Definicija sestavljene podatkovne strukture za opis posamezne celice.

Pri tem celoštevilčni atribut fire_stage opazovane celice $(x, y)$ predstavlja intenzivnost gorenja v njej na $k$-tem koraku, celoštevilčni atribut inflammability količino gorljivega materiala v celici na $k$-tem koraku in celoštevilčni atribut next_fire_stage intenzivnost gorenja v celici na koraku $k+1$. Pri tem vpeljemo tudi celoštevilčni atribut mehke spremenljivke wind_speed, ki definira hitrost vetra na požarnem področju, pri čemer zaradi enostavnosti predpostavimo, da je smer vetra konstantna in se veter giblje od leve proti desni glede na koordinatni sistem celic. Odtod lahko preidemo na definicije klasifikacij mehkih množic

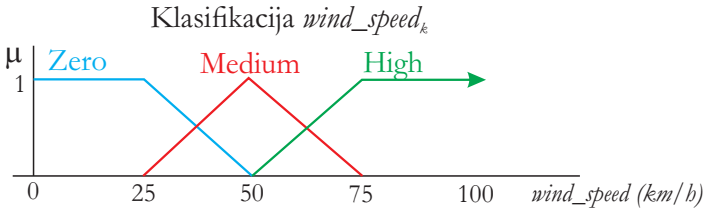
za navedene mehke spremenljivke. Grafično so ponazorjene na slikah 10.15, 10.16 in 10.17.




Slika 10.15: Definicija mehkih množic spremenljivke fire_stage.




Slika 10.16: Definicija mehkih množic spremenljivke inflammability.




Slika 10.17: Definicija mehkih množic spremenljivke wind_speed.

Po definiciji mehkih množic nastavimo vzorčni primer mehkega pravila za izračun novega stanja intenzivnosti gorenja v opazovani celici s koordinato $(x, y)$. Zaradi preglednosti pravila količine goriva v celici ne spreminjamo. Vzorčno pravilo predstavimo v izpisu 10.5.

```
1 if (cell[x,y].fire_stage=Medium) AND
2    (cell[x,y].inflammability=High) AND
3    (wind_speed=High) AND
4    (cell[x-1][y].fire_stage=High
5 then (cell[x,y].next_fire_stage=High);
```

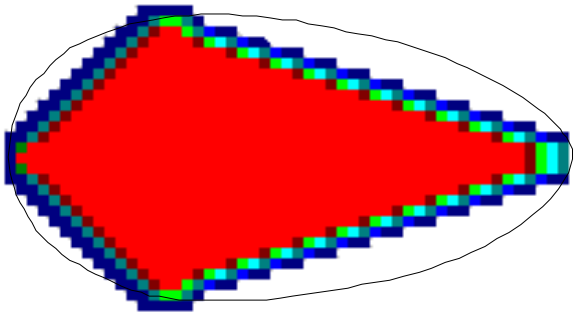
Listing 10.5: Vzorčno lingvistično pravilo programa posamezne celice modela.

V alineah v nadaljevanju opišemo karakteristike zasnovanega modela:

- podobno kot predhodno opisani mehki krmilnik tudi mehki program posamezne celice spreminja stanje posamezne celice v diskretnih časovnih korakih;
- model je bil zasnovan na opcijski povezavi z geografskim informacijskim sistemom Slovenije, pri čemer posamezna celica predstavlja površino $20m * 20m$ realnega terena;
- pred samim začetkom izvajanja simulacije moramo za vsako celico dvodimenzionalnega prostora določiti njeni začetni stanji intenzivnosti gorenja in količine gorljivega materiala; inicializiramo jih s prirejanjem celih števil oziroma ostrih vrednosti iz intervala $[0, 100]$; v prvem primeru ostro število predstavlja intuitivno oceno opazovalca o intenzivnosti gorenja v posamezni celici, v drugem primeru pa ostro število pridobimo avtomatizirano iz povezanega GIS sistema;
- vzorčno pravilo iz izpisa 10.5 upošteva intenzivnost gorenja zgolj v opazovani celici in njeni levi sosedi; tovrstno definicijo sosedstva smo uporabili zgolj zaradi preglednosti pravila;
- pogojni del vzorčnega pravila iz izpisa 10.5 omogoča formiranje 81 ($3^4 = 81$) različnih pogojnih delov pravil; vsak od pogojev namreč omogoča preverjanje pripadnosti ostre vrednosti k trem različnim mehkim množicam, preverjanje pa teče preko štirih pogojev; ob upoštevanju razširitve pravila na klasično Mooreovo sosednost, s čimer bi imela opazovana celica 8 sosedov, bi se nabor različnih pogojnih delov pravil razširil na 177.147 ($3^{11}$) različnih pogojnih delov pravil; iz omenjenega nabora lahko ob upoštevanju izkustvenih znanj eliminiramo pravila, ki so po naši subjektivni presoji nepotrebna;
- *vstopajoče ostre spemenljivke na k-tem koraku simulacije*: skozi simulacijo na $k$-tem diskretnem časovnem koraku za vsako celico v celični program vstopijo ostre vrednosti spremenljivk $fire\_stage$ in $inflammability$ ter ostra vrednost globalne spremenljivke $wind\_speed$;
- *mehčanje*: vse ostre vrednosti se zmehčajo - izračunajo se pripadnosti k definiranim mehkim množicam s slik 10.15, 10.16 in 10.17;
- *decizija*: postopek temelji na evaluaciji vseh pravil na vsaki celici v regularnem prostoru; sklepni deli pravil, v katerih nastopa mehka spremenljivka next_fire_stage, so bili pridobljeni s strani gasilskih strokovnjakov; definicija klasifikacij mehkih množic te spremenljivke je enaka definiciji klasifikacij mehkih množic spremenljivke fire_stage; končna verzija modela je uporabljala le 288 pravil od 177.147 možnih;
- *defuzifikacija*: postopek ostrenja mehkih vrednosti vseh celic je potekal na osnovi COG metode; po ostrenju izhodne spremenljivke next_fire_stage

se le ta priredi ostri vrednosti spremenljivke fire_stage na naslednjem časovnem koraku; slednje se izvede po celotnem prostoru celic;

Na slikah 10.18, 10.19 in 10.20 so prikazani simulacijski rezultati napovedi površine zajete s požarom in sicer po vrsti primerjava mehkega modela z Rothermelovim, primerjava Rothermelovega modela z obliko realnega požara in na koncu primerjava mehkega modela z obliko realnega požara. Iz slik lahko

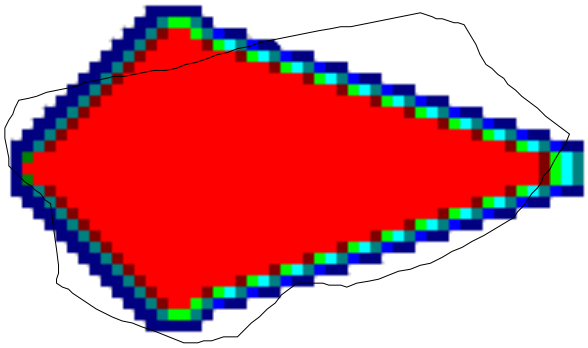


Slika 10.18: Primerjava napovedi mehkega modela z napovedjo Rothermelovega modela.

Slika 10.19: Primerjava napovedi Rothermelovega modela z obliko realnega požara.

razberemo, da uspešnost napovedi mehkega modela ni idealna (tako pri Rothermelovem modelu kot tudi v mehkem modelu ni bila upoštevana spremenljiva smer vetra), pri čemer pa so rezultati mehkega modela primerljivi z Rothermelovim. Prednost mehkega modela je v tem, da smo do njegove postavitve prišli relativno hitro in enostavno.

Po kategorizaciji metod v različne skupine naravnega procesiranja model širjenja požara na osnovi mehke logike sodi v drugo klasifikacijsko skupino.



Slika 10.20: Primerjava napovedi mehkega modela z obliko realnega požara.

# Literatura


[1] "The scale of things." `http://science.energy.gov/bes/community-resources/scale-of-things-chart/`, September 2016.

[2] M. Hak, *The MEMS Handbook*. CRC Press, 2002.

[3] M. Mack, "The multiple lives of Moore's law," *IEEE Spectrum*, vol. 4, pp. 29–35, 2015.

[4] D. Kodek, *Arhitektura in organizacija računalniških sistemov*. Bi-Tim, Slovenija, 2008.

[5] *From editors of Scientific American: Understanding nanotechnology*. Warner Books, ZDA, 2002.

[6] "2001: A Space Odyssey," 1968.

[7] A. Adamatzky, B. Costello, and T. Asai, *Reaction diffusion computers*. Elsevier, 2005.

[8] B. Hayes, "Third base," *American Scientist*, vol. 89, no. 6, 2001.

[9] W. Aspray, *John Von Neumann and The Origins Of Modern Computing*. The MIT Press, England, 1990.

[10] "There's Plenty of Room at the Bottom." `https://en.wikipedia.org/wiki/There%27s_Plenty_of_Room_at_the_Bottom/`, September 2016.

[11] E. Regis, *Nano – the emerging science of nanotechnology*. BackBay Books, 1995.

[12] C. Lent, P. Tougaw, W. Porod, and G. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, 1993.

[13] C. Lent and P. Tougaw, "Lines of interacting quantum-dot cells: a binary wire," *Journal of Applied Physics*, vol. 74, 1993.

[14] I. L. Bajec and M. Mraz, "Večstanjsko procesiranje v strukturah kvantnih celičnih avtomatov," *Elektrotehniški vestnik*, vol. 73, no. 2-3, 2006.

[15] P. Pečar, "Uporaba adiabatnega pristopa pri realizaciji trojiškega procesiranja na osnovi kvantnih celičnih avtomatov," Master's thesis, Faculty of computer and Information science, University of Ljubljana, 2007.

[16] G. Snider, A. Orlov, I. Amlani, and G. Bernstein, "Quantum-dot cellular automata: line and majority logic gate," *Japanese Journal of Applied Physics*, vol. 38, 1999.

[17] K. Walus, T. Dysart, G. Jullien, and R. Budiman, "Qcadesigner: a rapid design and simulation tool for quantum dots cellular automata," *IEEE Transactions on Nanotechnology*, vol. 3, 2004.

[18] T. Cole and J. Lusth, "Quantum-dot cellular automata," *Progress in Quantum Electronics*, vol. 25, 2001.

[19] Z. Kohavi, *Switching and finite automata theory*. McGraw-Hill Inc., USA, 1978.

[20] M. Niemier and P. Kogge, *Nano, Quantum and Molecular Computing - Implications to High Level Design and Validation*, ch. Origins and Motivations for Design Rules in QCA. Kluwer Academic Publishers, Boston, 2004.

[21] "Reverzibilnost procesiranja." `http://strangepaths.com/reversible-computation/2008/01/20/en/`, Oktober 2016.

[22] B. Hayes, "Reverse engineering," *American Scientist*, vol. 94, 2006.

[23] P. R. Yelekar and S. S. C. Hanson, "Introduction to reversible logic gates and its application," *International journal of computer applications*, 2011.

[24] S. M. R. Taha, *Reversible logic synthesis methodologies with application to quantum computing*. Springer, Switzerland, 2016.

[25] P. J. Denning and T. G. Lewis, "Computers that can run backwards," *American Scientist*, vol. 105, no. 5, 2017.

[26] "Reversible." `http://www.eng.fsu.edu/\symbol{126}mpf/pubs.htm`, Maj 2015.

[27] "Reversible." `http://www.cise.ufl.edu/research/revcomp/`, Maj 2015.

[28] K. Perumalla, *Introduction to Reversible Computing*. Chapman & Hall/CRC Press, 2014.

[29] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, UK, 2009.

[30] C. Calude and G. Păun, *Computing with cells and atoms, An introduction to quantum, DNA and membrane computing*. Taylor and Francis, London, 2001.

[31] "Single particle interference." http://player.slideplayer.com/26/8574538/#, December 2017.

[32] "Quantum computing." http://en.wikipedia.org/wiki/Quantum_computing, Maj 2015.

[33] M. Nagy and S. G. Akl, "Quantum computation and quantum information," tech. rep., 2005. Technical report, 2005-496.

[34] "Elementary quantum notation." http://www-users.cs.york.ac.uk/schmuel/comp/node6.html, Maj 2015.

[35] J. Virant, *Načrtovanje nanoračunalniških struktur.* Didakta, Slovenija, 2007.

[36] M. Hirvensalo, *Quantum Computing.* Springer Verlag, 2001.

[37] H. Stöcker, *Matematični priročnik z osnovami računalništva.* Tehniška založba, Slovenija, 2006.

[38] O. Bronštejn, K. Semendjajev, G. Musiol, and H. Mühlig, *Matematični priročnik.* Tehniška založba, Slovenija, 1997.

[39] A. Pittenger, *An Introduction to Quantum Computing Algorithms.* Birkhäuser, ZDA, 2000.

[40] D. Deutsch and R. Jozsa, "Rapid solutions of problems by quantum computation," *Proceedings of the Royal Society of London*, vol. 439, no. 1907, pp. 553–558, 1992.

[41] "Quantum programming languages." http://www.dcs.gla.ac.uk/~simon/quantum/, November 2015.

[42] "GUI environment for Quantum Computer Simulator." http://qcad.osdn.jp/, November 2015.

[43] "Quantiki homepage." http://www.quantiki.org/wiki/list-qc-simulators, November 2015.

[44] E. Dahl, "Programming with D-Wave: Map coloring problem," tech. rep., D-Wave Systems, 2013.

[45] "Radix economy." https://en.wikipedia.org/wiki/Radix_economy, October 2017.

[46] D. Miller and M. Thornton, *Multiple Valued Logic, Concepts and Representations.* Morgan and Claypool Publishers, USA, 2008.

[47] "Balanced ternary - Wikipedia." https://en.wikipedia.org/wiki/Balanced_ternary/, September 2016.

[48] J. Connelly, C. Patel, and A. Chavez, "Ternary Computing Testbed 3-Trit Computer Architecture," tech. rep., California Polytechnic State University of San Luis Obispo advised by Professor Phillip Nico, 2008.

[49] E. Katiyar, "A Treatise on the Fundamentals of Ternary Arithmetic and Logic," *International Journal of Computer Science Engineering*, vol. 6, no. 8, 2017.

[50] G. Trishala and K. Ragini, "Design and Implementation of Ternary Logic Circuits for VLSI Applications," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, 2020.

[51] "Ternary computer - Wikipedia." `https://en.wikipedia.org/wiki/Ternary_computer/`, September 2016.

[52] "Sistemska in sintezna biologija." `http://web.bf.uni-lj.si/bi/biokemija/SBD/Docs/sinbiol.pdf`, Maj 2015.

[53] "Sistemska biologija." `http://www.zrss.si/bzid/geni/pdf/baebler-clanek.pdf`, November 2017.

[54] "Deoksiribonukleinska kiselina." `https://hr.wikipedia.org/wiki/Deoksiribonukleinska_kiselina`, November 2017.

[55] M. Ridley, *Genom: biografija človeške vrste*. Učila International, 2002.

[56] J. D. Watson and A. Berry, *DNK - Skrivnost življenja*. Modrijan založba d.o.o., 2007.

[57] "Human Genome." `https://en.wikipedia.org/wiki/Human_genome`, November 2017.

[58] N. Cristianini and M.W.Hahn, *Introduction to Computational Genomics - A Case Study Approach*. Cambridge University Press, UK, 2007.

[59] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, pp. 1021–1024, 1994.

[60] L. M. Adleman, "Computing with DNA," *Scientific American*, vol. 279, pp. 54–62, 1998.

[61] C. S. Calude and G. Paun, *Computing with cells and atoms*. Taylor and Francis Inc., 2001.

[62] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, and et al., "The Sequence of the Human Genome," *Science*, vol. 291, pp. 1304–1351, Feb. 2001.

[63] J. C. Venter, *Genom mojega življenja*. Modrijan založba d.o.o., 2009.

[64] "Cost per genome." https://www.genome.gov/images/content/costpergenome_2017.jpg, November 2017.

[65] "DNA sequencing." https://en.wikipedia.org/wiki/DNA_sequencing, November 2017.

[66] "Cost to sequence human genome." https://en.wikipedia.org/wiki/DNA_sequencing, November 2018.

[67] M. Moškon, N. Zimic, and M. Mraz, "Realizacija dvojiškega pomnjenja v preprostih bioloških sistemih," *Elektotehniški vestnik*, vol. 83, no. 4, pp. 194–200, 2016.

[68] G. M. Church, Y. Gao, and S. Kosuri, "Next-Generation Digital Information Storage in DNA," *Science*, vol. 337, p. 1628, 2012.

[69] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, pp. 77–80, 2013.

[70] C. Bancroft, T. Bowler, B. Bloom, and C. Clelland, "Long-term storage of information in DNA," *Science*, vol. 293, no. 5536, pp. 1763–5, 2001.

[71] G. M. Church and E. Regis, *Regenesis: How Synthetic Biology Will Reinvent Nature and Ourselves.* Perseus Books Group, USA, 2012.

[72] M. Moškon, *Modeli in metrike dinamike preklopa v enostavnih bioloških sistemih za potrebe računalniških struktur prihodnosti.* PhD thesis, Univerza v Ljubljani, 2012.

[73] R. Gaber, T. Lebar, A. Majerle, B. Šter, A. Dobnikar, M. Benčina, and R. Jerala, "Designable dna-binding domains enable construction of logic circuits in mammalian cells," *Nature Chemical Biology*, vol. 10, no. 3, pp. 203–208, 2014.

[74] "Registry of standard biological parts." http://parts.igem.org/Main_Page, November 2017.

[75] M. Stražar, M. Mraz, N. Zimic, and M. Moškon, "An adaptive genetic algorithm for parameter estimation of biological oscillator models to achieve target quantitative system response," *Natural Computing*, vol. 13, pp. 119–127, 2014.

[76] "SBML Software Matrix." http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix, November 2017.

[77] "SBML Software Summary." http://sbml.org/SBML_Software_Guide/SBML_Software_Summary, November 2017.

[78] L. Borkowski, *Jan Łukasiewicz Selected works*. North-Holland Publishing Company, 1970.

[79] I. L. Bajec, N. Zimic, and M. Mraz, "Towards the bottom-up concept: extended quantum-dot cellular automata," *Microelectronic engineering*, vol. 83, no. 4/9, 2006.

[80] I. L. Bajec, N. Zimic, and M. Mraz, "The ternary quantum-dot cell and ternary logic," *Nanotechnology*, vol. 17, no. 8, 2006.

[81] P. Pečar, "Uporaba adiabatnega pristopa pri realizaciji trojiškega procesiranja na osnovi kvantnih celičnih avtomatov," Master's thesis, Faculty of computer and Information science, University of Ljubljana, 2007.

[82] P. Pečar, M. Mraz, N. Zimic, and I. L. Bajec, "Solving the ternary quantum-dot cellular automata logic gate problem by means of adiabatic switching," *Japanese journal of applied physics*, vol. 47, no. 6, 2008.

[83] P. Pečar, A. Ramšak, N. Zimic, M. Mraz, and I. L. Bajec, "Adiabatic pipelining: A key to ternary computing with quantum dots," *Nanotechnology*, vol. 19, no. 49, 2008.

[84] "Conway's Game of Life." `https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`, September 2016.

[85] S. Wolfram, *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.

[86] "Mathworld: Rule 30." `http://mathworld.wolfram.com/CellularAutomaton.html`, Oktober 2016.

[87] "Wikipedia - different rules evolution." `https://en.wikipedia.org/wiki/Elementary_cellular_automaton#Rule_28`, Oktober 2016.

[88] "Elementary cellular automata." `http://mathworld.wolfram.com/ElementaryCellularAutomaton.html`, Oktober 2018.

[89] M. Sipper, *Evolution of Parallel Cellular Machines*. Springer Inc., UK, 1997.

[90] "Orodja za simulacije evolucije CA." `http://uncomp.uwe.ac.uk/genaro/Cellular_Automata_Repository/Software.html`, Oktober 2016.

[91] "Manxiu Zhan: Amorphous computing." `https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/2013WS-HS-AmorphousComputing.pdf`, January 2014.

[92] "Amorphous computing." `http://groups.csail.mit.edu/mac/projects/amorphous/white-paper/amorph-new/amorph-new.html`, Maj 2015.

[93] H. Abelson, J. Beal, and G.J.Sussman, "Amorphous computing," tech. rep., 2007. MIT, Technical Report.

[94] H. Abelson, D. Allen, D. Coore, and C. Hanson, "Amorphous computing," *Communications of the ACM*, vol. 94, no. 5, 2000.

[95] "Amorphous computing." `http://groups.csail.mit.edu/mac/projects/amorphous/paperlisting.html\#daniel-thesis`, Maj 2015.

[96] "GPL inverter." `https://groups.csail.mit.edu/mac/projects/amorphous/workshop-sept-99/`, November 2016.

[97] "What is language?." `http://www.cs.virginia.edu/\symbol{126}evans/cs655-S00/lectures/lecture23.ppt\#380,42,Whatisalanguage?(Lecture1)`, Maj 2015.

[98] "Programming for swarms." `www.cs.virginia.edu/~evans/cs655/projects/zhong.ppt`, November 2016.

[99] "Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics." `https://pdos.csail.mit.edu/~micahbro/junk/nagpal-thesis%5B2%5D.pdf`, December 2017.

[100] J. Virant, *Z računalnikom v fraktalsko geometrijo narave*. Didakta, Slovenija, 1991.

[101] J. Virant, *Uporaba mehke logike v sodobnih sistemih*. Didakta, Slovenija, 1992.

[102] M. Mraz, N. Zimic, and J. Virant, "Intelligent bush fire spread prediction using fuzzy cellular automata," *Journal of Intelligent and Fuzzy Systems*, pp. 203–207, 1999.