**Algoritamske heuristike** 22.04.2024. (aprilski rok 2024.)

*Napomena: ispitni zadaci se rade samostalno na računaru.*
*Rok za predaju radova (kodovi, mini izveštaj/prezentacija sa prikazom rezultata): tri nedelje, ponedeljak 13. maj 2024. u 12h. Nakon toga ćemo organizovati sastanak preko MS Teams-a na kome ćete kratko prikazati i odbraniti svoje radove.*

1. (**Rešavanje Sudoku metaheuristikom simuliranog kaljenja**) Problem rešavanja $n^2 \times n^2$ Sudoku ukrštenice (ili tzv. Number Placement problem) je **NP**-kompletan (vidite priloženi članak "Compl and Completeness of Finding ASP and Its Appl to Puzzles.pdf "). U najrasprostranjenijoj verziji ove igre ($n$=3) treba rasporediti brojeve od 1 do 9 u 9×9 kvadrata (ćelija), podeljenih (ili grupisanih) u devet 3×3 kvadrata, koji se nazivaju regioni. Neki od 9×9 kvadrata su delimično već popunjeni sa datim brojevima (eng. "givens"). Preostale prazne kvadrate treba popuniti tako da se u svakoj vrsti, koloni i regionu brojevi od 1 do 9 javljaju tačno jedanput.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | | | | 1 | 4 | | |
| 2 | | 3 | | | | 7 | | |
| | 7 | | 3 | | | 1 | 8 | 2 |
| | | 4 | | 5 | | | | 7 |
| | | | 1 | | 3 | | | |
| 8 | | | | 2 | | 6 | | |
| 1 | 8 | 5 | | | 6 | | 9 | |
| | | 2 | | | | 8 | | 3 |
| | | 6 | 4 | | | | 7 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 4 | | 5 | | | 6 | |
| | 6 | | 1 | | | 8 | | 9 |
| 3 | | | | | 7 | | | |
| | 8 | | | | | 5 | | |
| | | | 4 | | 3 | | | |
| | | 6 | | | | 7 | | |
| | | | 2 | | | | | 6 |
| 1 | | 5 | | | 4 | | 3 | |
| | 2 | | | 7 | | 1 | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | | | 6 | 7 | | | | |
| | | 6 | | | | 2 | | 1 |
| 4 | | | | | | 8 | | |
| 5 | | | | | 9 | 3 | | |
| | 3 | | | | | | 5 | |
| | | 2 | 8 | | | | | 7 |
| | | 1 | | | | | | 4 |
| 7 | | 8 | | | | 6 | | |
| | | | | 5 | 3 | | | 8 |

Slika 1. lak primer Sudoku srednje težak primer težak primer

U ovom zadatku se traži da implementirate i testirate algoritam simuliranog kaljenja (eng. Simulated Annealing, kratko SA) za rešavanje Sudoku ukrštenice, bazirano na priloženom članku "Metaheuristics can Solve Sudoku Puzzles.pdf" i implementiranom SA algoritmu koji možete skinuti sa adrese http://yarpiz.com/223/ypea105-simulated-annealing . Ulaz za vaš algoritam je delimično popunjena Sudoku ukrštenica. Vaš kod treba da zadovolji sledeće zahteve:

- ***Inicijalno rešenje:*** Pseudo-slučajno popunite svaki od praznih kvadrata unutar 3×3 regiona, tako da svi regioni sadrže sve brojeve od 1 do 9 tačno jedanput. Drugim rečima, u inicijalnom rešenju ne morate voditi računa da se u vrstama i kolonama pojavljuju brojevi od 1 do 9 tačno jedanput. Koraci koji slede u algoritmu nikada neće menjati frekvenciju pojavljivanja brojeva od 1 do 9 u svakom 3×3 regionu (tj. uvek će ih biti po devet različitih u svakom regionu).
- Svako (potencijalno) rešenje koje će biti uzeto u razmatranje tokom pretrage (tokom rada algoritma) će u svakom 3×3 regionu sadržati tačno jedan od brojeva od 1 do 9. Vrste i kolone ne moraju da zadovoljavaju taj zahtev.
- ***Funkcija cene koju treba minimizovati***: Za svaku vrstu $i$ (kolonu $j$) vrednost vrste $rs_i$ (vrednost kolone $cs_j$, respektivno) je broj simbola (tj. brojeva od 1 do 9) *koji nedostaju* u vrsti $i$ (koloni $j$, respektivno). Ukupna cena nekog potencijalnog rešenja je suma svih vrednosti vrsta i vrednosti kolona, odnosno $\left(\sum_{i=1}^{9} rs_i\right)+\left(\sum_{j=1}^{9} cs_j\right)$. Očigledno, rešenje Sudoku ukrštenice će biti ono čija je ukupna cena 0. Pogledajte primer računanja cene u priloženom članku "Metaheuristics can Solve Sudoku Puzzles.pdf".
- ***Definisanje susedstva u prostoru pretrage:*** Svaki sused trenutnog rešenja se dobija operacijom zamene (eng. swapping) vrednosti dve ne-fiksirane ("non-given") ćelije koje pripadaju istom 3×3 regionu. Pretraga susedstva sledi paradigmu simuliranog kaljenja – tako da se prvo pseudo-slučajno izabere jedna ne-fiksirana ćelija; zatim se pseudo-slučajno izabere druga ne-fiksirana ćelija u istom regionu, i njihove vrednosti se zamene.

a) Napišite .m kod za implementaciju algoritma simuliranog kaljenja koji rešava ovaj problem. Koristite kostur SA algoritma dat na adresi http://yarpiz.com/223/ypea105-simulated-annealing. Pogledajte i video tutorijal na istom sajtu, http://yarpiz.com/440/ytea101-particle-swarm-optimization-pso-in-matlab-video-tutorial . Sve metaheuristike na yarpiz.com sajtu su implementirane na isti način, i lako ih je prilagoditi konkretnom problemu koji se rešava (praktično treba samo iskodirati instance problema, susedstvo i funkciju cilja).

b) Eksperimentišite da biste našli odgovarajuće vrednosti za inicijalnu temperaturu $t_0$, faktor hlađenja $\alpha$ (autor članka predlaže $\alpha$=0.99) i maksimalni broj iteracija $c_{max}$. Kriterijum zaustavljanja treba da bude i) kada je ukupna cena 0 ili ii) kad se dostigne maksimalan broj iteracija $c_{max}$, tako da $c_{max}$ postavite da bude dovoljno veliko, u nadi da ga nikada ne dostignete.

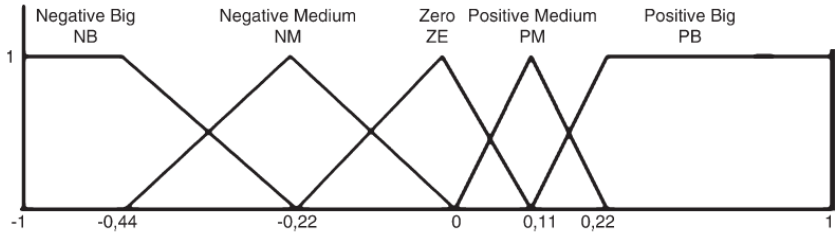
c) Testirajte svoj algoritam na test-primerima priloženim u datoteci "sudoku_instance.zip", u kojoj se nalazi 11 lakih instanci (datoteke "L1.txt"..."L11.txt"), 11 srednje teških instanci (datoteke "S1.txt"..."S11.txt") i 11 teških instanci (datoteke "T1.txt"..."T11.txt"). U ovim datotekama, zadati brojevi su označeni sa {1,2,...,9}, a nedostajući brojevi sa nulama. (Napomena: u primerima teških instanci problema T1.txt-T11.txt iz datoteke sudoku_instance.zip ima grešaka; možete koristiti sudoku_instance_2.zip koje su preuzete sa stranice lipas.uwasa.fi/~timan/sudoku i takođe priložene ovde).

2. **(Intervalna Fazi logika (Type 2 Fuzzy Logic) i Fazi logički PID kontroleri (FLC) tipa 1 i tipa 2)** Ovaj zadatak je posvećen fazi logici "drugog tipa". Svi primeri iz fazi logike sa kojima smo se upoznali tokom AH kursa školske 2023/24 godine su bili "prvog tipa/prve vrste". Upoznajte se (samo) sa osnovama fazi logike drugog tipa iz priložene literature (knjige i članci) koje se nalaze u datoteci "Uz_zadatak_2.zip".

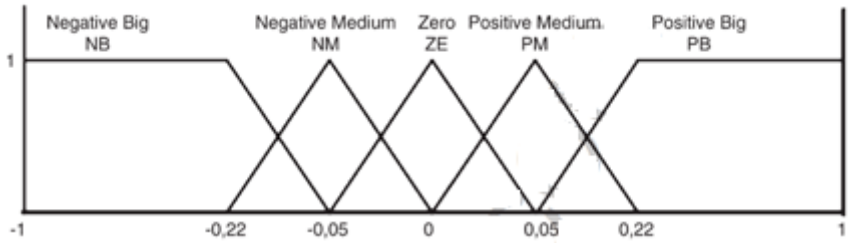
Počev od verzije R2019b, MatLab podržava rad sa type-2 fuzzy logikom, tako što može da konvertuje fazi sisteme tipa 1 u tip 2 (pogledajte npr. primer sa web-strane https://nl.mathworks.com/help/fuzzy/fuzzy-pid-control-with-type-2-fis.html ). U ovom zadatku se ne traži da koristite tu mogućnost, već slobodno dostupan toolbox koji se nalazi u zip datoteci "type-2-fuzzy-logic-systems-matlab-toolbox-1.1.zip", autora A. Taskina i T. Kumbaskara. U njemu je implementiran Fuzzy Inference Systems (FIS) editor za Type-2 fazi logiku, koji je veoma sličan MatLab-ovom FIS editoru koji smo koristili na vežbama. U datoteci "taskin2015 type2 fuzzy.pdf" je opisan ovaj editor, koji se poziva sa *fuzzyt2*. U okvirima Type-2 toolbox-a i njegovog pratećeg opisa je dat i primer Type-2 Fuzzy PID kontrolera (pokreće se sa "runControlSystemExample.m", a njegovi detalji se mogu videti u datotekama "IT2_FPID.t2fis" koja se otvara u *fuzzyt2* editoru, i "IT2_FPID_Controller.slx" modelom u Simulink-u).

a) Dobro se upoznajte sa primerom fazi logičke kontrole tipa-2 i detaljno pročitajte članak A. Taskina i T. Kumbaskara. Napišite mini-izveštaj (jedna do dve A4 strane na kojima ćete opisati najosnovnije pojmove iz fazi logike drugog tipa, i glvne razlike u odnosu na fazi logiku tipa 1).

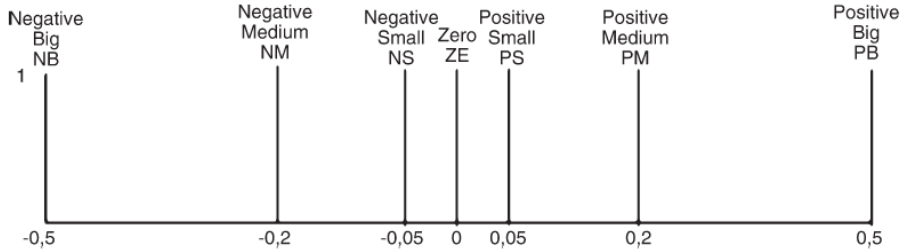
b) Implementirajte Sugeno FIS tipa 1 koji ima dve ulazne fazi varijable koje se zovu „*e*" (od „error"), i „*de*" (od „error derivate", izvod greške po vremenu, ili brzina greške, ili promena greške u vremenu) čije su membership funkcije (MF-je) zadate na slici 1 i slici 2, jednu izlaznu „krisp" (crisp, oštru, non-fuzzy) izlaznu varijablu (slika 3), i 25 pravila (slika 4). Pravila su ista i za „*e*" i za „*de*". Koristite MatLab-ov FIS editor za sisteme tipa-1 (poziva se iz komandnog prompta naredbom *fuzzy*).




Slika 1. MF-je za grešku *e* (prva ulazna fazi promenjiva tipa 1).




Slika 2. MF-je za promenu greške *de* (druga ulazna fazi promenjiva tipa 1).



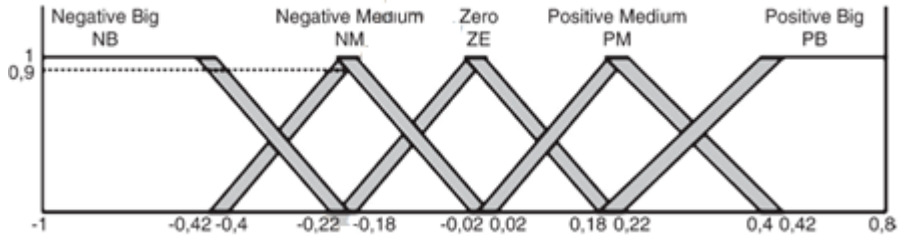

Slika 3.MF-je za izlaznu promenjivu Sugeno FIS sistema tipa 1.

Table 2
Rule base for type-1 and type-2 FLC

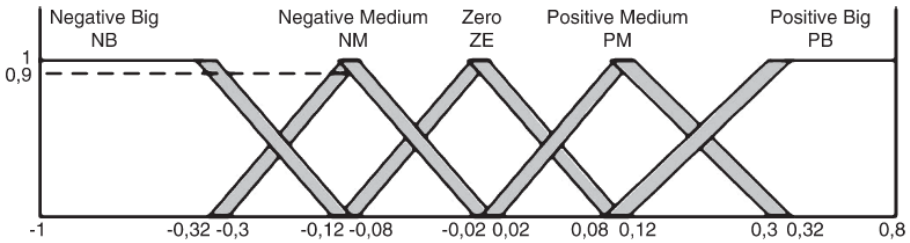
| e / de | NB | NM | ZE | PM | PB |
|---|---|---|---|---|---|
| NB | NB | NB | NM | NS | ZE |
| NM | NB | NM | NS | ZE | ZE |
| ZE | NM | NS | ZE | ZE | PS |
| PM | ZE | ZE | PS | PM | PB |
| PB | ZE | PS | PM | PB | PB |

Slika 4. Pravila (Rule base, RB) za Sugeno FIS tipa 1 i tipa 2.

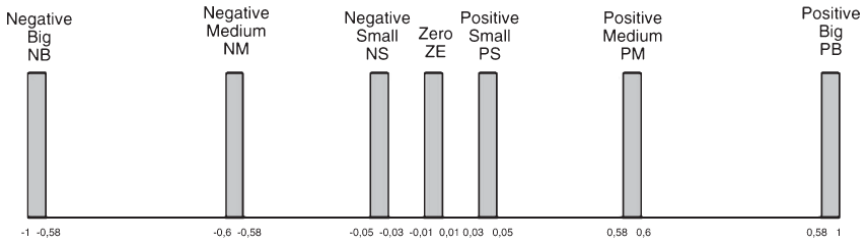
c) Implementirajte Sugeno FIS tipa 2 koji ima dve ulazne fazi varijable koje se zovu „*e*“ i „*de*“, čije su membership funkcije (MF-je) zadate na slici 5 i slici 6, jednu intervalnu izlaznu varijablu čije su membership funkcije zadate na slici 7, i istih 25 pravila kao pod b) (slika 4).




Slika 5. MF-je za grešku *e* (prva ulazna fazi promenjiva tipa 2).




Slika 6. MF-je za promenu greške *de* (druga ulazna fazi promenjiva tipa 2).




Slika 7.MF-je za izlaznu promenjivu Sugeno FIS sistema tipa 2. Napomena: na slici 7 su dve greške, interval za NB je [– 1, – 0.98], interval za PB je [0.98, 1]. Vrednost svih intervalnih MF-ja je 1.

d) Uvucite dva FIS-a koja ste razvili (FIS Tipa-1 pod b) i FIS Tipa-2 pod c) ) u FLC PID kontroler dat u priloženom toolbox-u, nacrtajte grafik kako izlazni signal prati jedinični ulazni (referentni) signal, i uporedite FLC PID kontroler Tipa-1 i Tipa-2.

3. **(Soft computing - Veštačke neuralne mreže)**

a) Koristeći MatLab-ov Neural Network Toolbox (koji pokrećete naredbama *nnstart* ili *nntool*), kreirajte/istrenirajte veštačku neuralnu mrežu (ANN, Artificial Neural Network) koja će implementirati funkcionalnost 2-na-1 multipleksera (dva ulazna signala, jedan selekcioni signal i jedan izlazni signal).

b) koristeći Bulovo kolo koje ste napravili pod a), implementirajte u Simulink-u Bulovu funkciju (BF) $y = f(a,b,c,d) = \overline{a} \cdot \overline{c} \cdot \overline{d} + b \cdot \overline{c} \cdot \overline{d} + a \cdot \overline{b} \cdot \overline{c} + \overline{a} \cdot b \cdot \overline{c} \cdot d + \overline{a} \cdot \overline{b} \cdot c \cdot \overline{d}$ nad funkcionalno potpunim bazisom {MUX}. Napomena: ovde vam može biti od pomoći minimizacija BF preko ROBDD-ova i tzv. Šenon-ova dekompozicija BF, koju možete dobiti npr. korišćenjem alata JADE (http://www.informatik.uni-bremen.de/agra/eng/jade.php). Implementaciju BF nad bazisom {MUX} dobijate ako svaki čvor u dobijenom ROBDD-u zamenite sa po jednim MUX-om.

c) Ponavljajući postupak pod a) i b), istrenirajte ANN koja će implementirati funkcionalnost 4-na-1 MUX-a i implementirajte istu BF zadatu pod b) koristeći 4-na-1 MUX-eve.