RI odgovori na pitanja

Mina Milošević
mi17081@alas.matf.bg.ac.rs

2020/2021

Materijal je preuzet sa stranice prof. Aleksandra Kartelja.

Sadržaj

1 Uvod u računarsku inteligenciju

1.1 Računarska inteligencija, definicija i paradigme

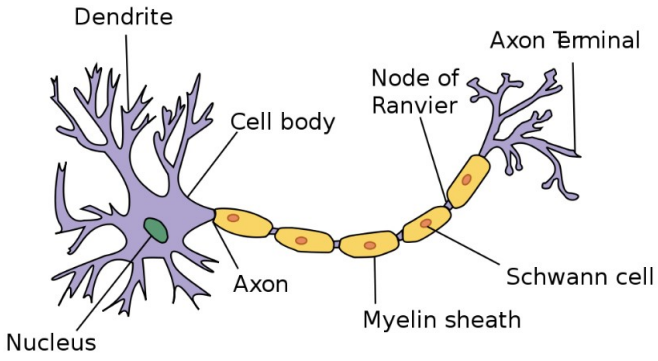
Inteligencija se definiše kao mogućnost razumevanja i ostvarivanja koristi od razumevanja. Uključuje kreativnost, veštine, svesnost, emocije i intuiciju.
Računarska inteligencija (eng. Computation intelligence - CI) je podgrana Veštačke inteligencije (eng. Artificial intelligence - AI). Izučava mehanizme inteligentnog ponašanja u složenim i promenljivim okruženjima. Mehanizmi koji mogu da uče, da se prilagođavaju, uopštavaju, itd.
Paradigme:

- Veštačke neuronske mreže(eng. Artificial neural networks - ANN)
- Evolutivna izračunavanja (eng. Evolutionary computation - EC)
- Inteligencija grupa (eng. Swarm intelligence - SI)
- Veštački imuni sistem (eng. Artificial immune system - AIS)
- Rasplinuti (fazi) sistemi (eng. Fuzzy systems - FS)

Srodan termin *CI* je Soft computing koji podrazumeva upotrebu višestrukih *CI* paradigmi i verovatnosnih metoda.

1.2 Veštačke neuronske mreže, definicija, biološki neuron, veštački neuron

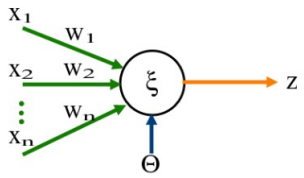
Ljudski mozak: prepoznavanje oblika; percepcija; motorika; mnogo efikasniji u tim zadacima nego bilo koji računar (trenutno); između 10 i 500 milijardi neurona i 60 triliona sinapsi. Da li će ikada biti moguće modelovati ga u potpunosti?
Današnje veštačke neuronske mreže (ANN) omogućavaju rešavanje pojedinačnih zadataka; mozak može da rešava mnogo zadataka istovremeno.



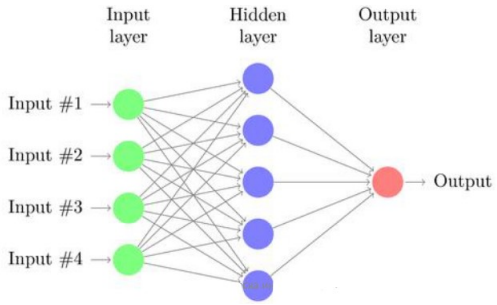
Biološki neuron

Biološki neuron - neuroni su vrlo povezani među sobom. Veza se ostvaruje između aksona jednog neurona i dendrita drugog – ova veza se zove sinapsa. Signali se kreću od dendrita, preko tela ćelije do aksona. Slanje signala se dešava povremeno – kada ćelija „ispali“. Neuron može da suzbije ili pojača jačinu signala.

Veštački neuron (eng. Artificial neuron – AN) - model biološkog neurona. AN prima signal od okruženja ili drugog AN. Nakon toga, on ga prenosi povezanim AN. Ispaljivanje signala i njegova jačina su kontrolisani formulom.



Veštačka neuronska mreža (ANN) - mreža sa slojevito poređanim AN. Obično sadrži ulazni, izlazni i nula ili više središnjih slojeva.
Vrste ANN: jednoslojne, višeslojne mreže sa propagacijom unapred, temporalne i rekurentne, samoorganizujuće, kombinovane mreže.
Primene ANN: medicinska dijagnostika, prepoznavanje zvuka, procesiranje slika, predviđanje u analizi vremenskih serija, kontrola robota, klasifikacija podataka, kompresija podataka...



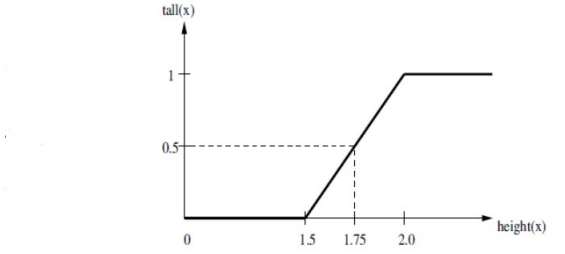
2 Fazi sistemi

2.1 Uvod u fazi sisteme i fazi skupovi

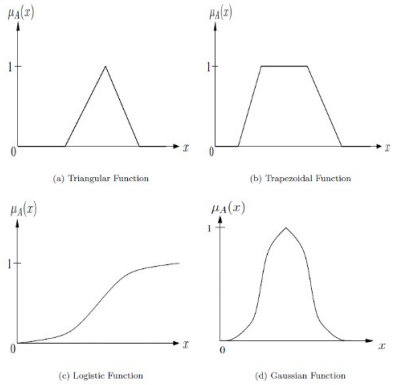
Klasična binarna logika nije često primenljiva u rešavanju realnih problema. Realni problemi se opisuju često neprecizno, nekompletno (npr. rečenica „delimično je oblačno“ nije binarna). Kako zaključivati na osnovu ovakvih izjava? Potrebno je razviti drugačiji logički sistem. Za te potrebe, kreće se od drugačije definicije skupova i funkcije pripadnosti skupu. (Lofti Zadeh - 1965)
Kako definisati skup visokih ljudi? U klasičnoj teoriji skupova, na osnovu neke granice, npr. 175cm. Problem sa ovim je što je i osoba od 178cm i 210cm samo „visoka“. Takođe postoji oštra granica na prelazima, tj. u blizini granice.
Fazi skupovi omogućavaju da se pripadnost skupu definiše sa nekom numeričkom vrednošću koja je između 0 i 1. Ako je X domen, a $x \in X$ konkretni element tog domena., onda se fazi skup A opisuje funkcijom pripadnosti: $\mu_A : X \to [0, 1]$. Fazi skupovi mogu biti definisani nad diskretnim ili realnim domenima.
Postoje dve notacije za predstavljanje diskretnog fazi skupa:

- Preko skupa uređenih parova: $A = \{(\mu_A(x_i), x_i) | x_i \in X, i = 1, ..., n_x\}$
- Preko „sume“: $A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + ... + \mu_A(x_{nx})/x_{nx}$. Oprez: ova suma nije prava, ne podrazumeva stvarno sabiranje, već je samo vid redefenisane notacije koja je u saglasnosti sa notacijom za realni domen.

Notacija za realni fazi skup je data preko „integrala“: $A = \int_X \mu(x)/x$. Opet ovo nije pravi „integral“ već samo pogodna notacija.
Fazi funkcija pripadnosti skupu - ograničena između 0 i 1. Za svaki element domena je jednoznačna. Na slici je prikazan jedan mogući način definisanja skupa visokih ljudi i standardne fazi funkcije pripadnosti.



$$tall(x) = \begin{cases} 0 & \text{if } length(x) < 1.5m \\ (length(x) - 1.5m) \times 2.0m & \text{if } 1.5m \leq length(x) \leq 2.0m \\ 1 & \text{if } length(x) > 2.0m \end{cases}$$



2.2 Fazi skupovne operacije

- *Jednakost skupova* - fazi skupovi su jednaki ako imaju isti domen i pritom za svaki element domena imaju istu funkciju pripadnosti: $A = B$ akko $\mu_A(x) = \mu_B(x)$ za sve $x \in X$
- *Podskupovi* - skup A je podskup skupa B akko $\mu_A(x) \leq \mu_B(x)$ za sve $x \in X$
- *Komplement* - ako je A^C komplement skupa A, onda za sve $x \in X$, $\mu_A(x) = 1$ - $\mu_{A^C}(x)$. Ne važi identiteti kao u klasičnoj teoriji skupova da je $A^C \cap A = \emptyset$ i $A^C \cup A = X$.
- *Presek* - postoji mnogo načina, a standardni su:
  - preko minimuma: $\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \forall x \in X$
  - preko proizvoda: $\mu_{A \cap B}(x) = \mu_A(x)\mu_B(x), \forall x \in X$. Sa proizvodom treba biti oprezniji: već nakon nekoliko uzastopnih primena funkcije pripadnosti teže 0.
- *Unija* - postoji mnogo načina, a standardni su:
  - preko maksimuma: $\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\}, \forall x \in X$
  - preko sume i preseka: $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x), \forall x \in X$. Sa sumacijom i preskom treba biti oprezniji, jer funkcije pripadnosti teže 1 čak iako su polazne funkcije bliske 0.

2.3 Karakteristike fazi skupova

- *Normalnost* - fazi skup A je normalan ako ima bar jedan element koji pripada skupu sa stepenom 1: $\exists x \in A \cdot \mu_A(x) = 1$ ili $sup_x \mu_A(x) = 1$
- *Visina* - supremum po funkciji pripadnosti: $height(A) = sup_x \mu_A(x)$
- *Podrška* - skup svih elemenata koji imaju pripadnost veću od 0: $support(A) = \{x \in X | \mu_A(x) > 0\}$
- *Jezgro* - skup svih elemenata koji pripadaju skupu sa stepenom 1: $core(A) = \{x \in X | \mu_A(x) = 1\}$
- *α-rez* - skup svi elemenata koji imaju pripadnost najmanje α: $A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\}$
- *Unimodalnost* - fazi skup je unimodalan ako njegova funkcija pripradnosti unimodalna
- *Kardinalnost* - u zavisnosti od tipa domena, definiše se kao:
  - $card(A) = \sum_{x \in X} \mu_A(x)$
  - $card(A) = \int_{x \in X} \mu_A(x) dx$
- *Normalizacija* - fazi skup se normalizuje tako što se funkcija pripadnosti podeli visinom fazi skupa: $normalized(A) = \mu_A(x) - height(x)$
- Ostala bitna svojstva su: *komutativnost*, *asocijativnost*, *tranzitivnosti* i *idempotencija*.

2.4 Fazi i verovatnoća

Postoji česta zabuna između fazi koncepta i verovatnoće. Oba termina referišu na (ne)sigurnost događaja, ali tu sve sličnosti prestaju.
Statistička verovatnoća se vezuje za posmatranje događaja koji treba tek da se dese i za donošenje zaključaka o tome kolika je njihova šansa da će se desiti. Ako posmatramo uzastopna bacanja novčića i na velikom broju bacanja utvrdimo da je u 50% situacija „pala“ glava, tada možemo govoriti o sigurnosti toga da će u narednom bacanju da „padne“ glava. Kod fazi skupova, funkcija pripadnosti ne govori o sigurnosti da se neki događaj iz budućnosti desi. Npr. imamo dve osobe, jedna je visoka 220cm i pripada skupu visokih ljudi sa stepenom 1 i imamo drugu koja je visoka 195cm i pripada skupu visokih ljudi sa stepenom 0.95. Interpretacija ne znači da će u ponovljenim eksperimentima prva osoba biti češće veća od druge osobe. To samo znači da su obe osobe visoke ali sa različitim stepenom pripadnosti skupu visokih osoba. Dakle, fazi se vezuje za stepen istinitosti, dok se verovatnoća vezuje za mogućnost predviđanja nekog ishoda.

2.5 Fazi logika

Ključni elementi: lingvističke promenljive i fazi if-then pravilo zaključivanja.
Lingvistička (fazi) promenljiva –Zadeh (1973) - promenljive čije su vrednosti reči prirodnog jezika (npr. reč tall je lingvistička promenljiva). Tipovi lingivističkih promenljivih:

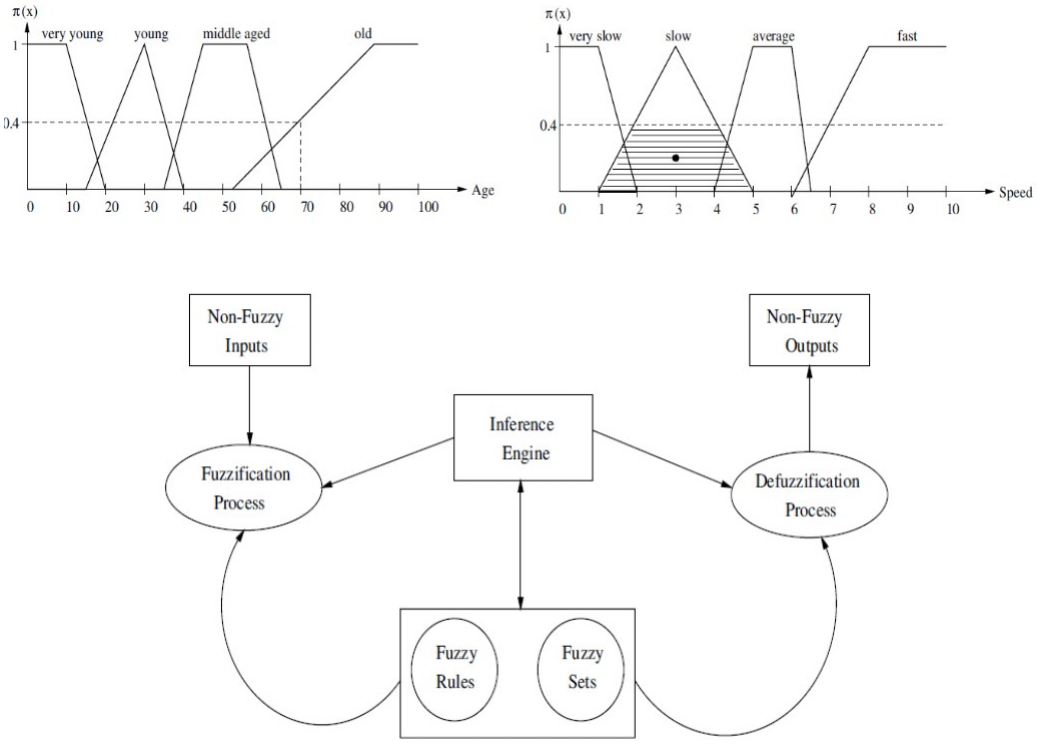
- kvantifikatori: sve, većina, mnogo, nijedan, itd.
- promenljive za učestalost: ponekad, često, uvek, itd.
- Promenljive za šansu: moguće, verovatno, sigurno, itd.

Modifikatori su dodatne reči koje pojačavaju ili slabe efekat lingvističkih promenljivim. Najčešće su u pitanju neki pridevi, npr. veoma, malo, srednje, itd. Mogu biti dovedeni u relaciju sa originalnom lingvističkom promenljivom putem funkcije.
Primer: $\mu_{verytall}(x) = \mu_{tall}(x)^2$. Ako neko pripada skupu visokih sa sigurnošću 0.9, onda će pripadati skupu veoma visokih sa manjom sigurnošću od 0.81.
Modifikatori za pojačavanje obično imaju formu: $\mu_{A'}(x) = \mu_A(x)^{1/p}$ za p > 1.

2.6 Fazi zaključivanje

Primer:
$\mu_{tall}(Peter) = 0.9$ i $\mu_{goodathlete}(Peter) = 0.8$
$\mu_{tall}(Carl) = 0.9$ i $\mu_{goodathlete}(Carl) = 0.5$
Ako se zna da je dobar košarkaš visok i atleta, koji od ove dvojice je bolji? Primenom pravila minimuma za presek (logičko i):
$\mu_{goodbasketballplayer}(Peter) = min\{0.9, 0.8\} = 0.8$
$\mu_{goodbasketballplayer}(Carl) = min\{0.9, 0.5\} = 0.5$
Pa zaključujemo da je Peter bolji košarkaš. U realnim okolnostima, zavisnosti su mnogo složenije pa govorimo o skupu if-then pravila.

Primer: if Age is Old the Speed is Slow. Na osnovu skupa premisa se donose skup zaključaka. Pripadnost osobe od 70 godina starim osobama je 0.4. Centar gravitacije u oblasti spore brzine ograničenoj sa 0.4 je 3.



Mamandijev metod

Fazifikacija - ulazni podaci (premise) se iz ulaznog prostora (koji nije fazi) prevode u fazi reprezentaciju. Primena funkcije pripadnosti nad ulaznim podatkom. Npr. ako su A i B fazi skupovi nad domenom X, proces fazifikacije prihvata elemente $a, b \in X$. Na izlazu proizvodi fazi skup tako što im dodeljuje stepene pripadnosti svakom od fazi skupova: $\{(\mu_A(a), a), (\mu_B(a), a), (\mu_A(b), b), (\mu_B(b), b)\}$.
Primena pravila zaključivanja:
Cilj je primeniti pravila zaključivanja nad fazifikovanim ulazima. Na izlazu iz pravila zaključivanja je fazifikovani izlaz za svako od pravila. Neka su A i B definisani nad domenom X_1, dok je fazi skup C definisan nad domenom X_2. Neka je pravilo: *if A is a and B is b then C is c*. Na osnovu fazifikacije znamo: $\mu_A(a)$ i $\mu_B(b)$. Prvo je potrebno izračunati stepen pripadnosti skupu premisa: $min\{\mu_A(a), \mu_B(b)\}$. Ovo se radi za svako od pravila zaključivanja. Neka je α_k stepen pripadnosti premisa za k-to pravilo. Sledeći korak je računanje stepena pripadnosti zaključku c_i: $\beta_i = max\{\alpha_{ki}\}$, za svako pravilo k u kojem figuriše c_i. To znači da je na izlazu iz zaključivanja stepen pripadnosti za svaki od fazi skupova zaključaka.
Poslednji korak je prevođenje fazi zaključaka u ne-fazi (defazifikacija). Ovo podrazumeva određivanje lingvističkih promenljivih za prethodno određenje stepene pripadnosti zaključcima. Pristupi zasnovani na računanju centroide:

- $output = \frac{\sum_{i=1}^{n_x} x_i \mu_C(x_i)}{\sum_{i=1}^{n_x} \mu_C(x_i)}$
- $output = \frac{\int_{x \in X} x \mu_(x) dx}{\int_{x \in X} \mu_(x) dx}$

Nakon računanja centroide, pročita se ona lingvistička promenljiva koja joj odgovara prema nekom od sledećih pravila:

- *max-min* – uzima se centroida ispod lingvističke promenljive koja odgovara zaključku sa najvišim stepenom, u ovom slučaju je to LI
- *uprosečavanje* – računa se centroida za sve lingvističke promenljive i na osnovu toga određuje konačna lingvistička promenljiva
- *skaliranje* – funkcije pripadnosti se skaliraju prema dobijenim zaključcima i nakon toga se računa centroida
- *isecanje* – funkcije pripadnosti se seku na mestima koja odgovaraju zaključcima i potom se računa centroida

3 Optimizacija

3.1 Optimizacija, definicija, izazovi, ključni pojmovi

Optimizacioni algoritmi pripadaju grupi algoritama pretrage. Cilj je pronaći rešenje problema takvo da je:

1. Neka ciljna funkcija (*funkcija cilja*) maksimalna/minimalna
2. Neki *skup ograničenja* zadovoljen (opciono)

Izazovi:

- Rešenje može biti predstavljeno kao kombinacija vrednosti iz različitih domena
- Ograničenja mogu biti nelinearna
- Karakteristike problema mogu varirati tokom vremena
- Funkcija cilja može biti u „konfliktu“ sa ograničenjima

Ključni pojmovi u optimizaciji:

- Neka je sa $f : S \to R$, označena funkcija cilja
- S je domen, dok je kodomen skup realnih brojeva
- Primetiti da je minimum f isto što i maksimum od $-f$
- Skup x predstavlja nezavisne promenljive koje utiču na vrednost f i za date vrednosti promenljivih funkcija ima vrednost *f(x)*
- Skup ograničenja najčešće postavlja zavisnosti između nezavisnih promenljivih.
- Takođe omogućava i ograničavanje samih nezavisnih promenljivih, npr. na neki interval ili skup vrednosti.

Postoje oblast koja se zove *programiranje ograničenja* (eng. Constraint programming). Ona se bavi problemima bez funkcije cilja. Ovde postoje samo ograničenja koja treba zadovoljiti. Optimizacija se ne bavi ovakvim problemima! Iako se nekim optimizacionim metodama mogu rešavati i ovakvi problemi.
Tipovi optimuma:

- *Globalni optimum* - najbolje rešenje na čitavom dopustivom skupu rešenja S, $f(x^*) < f(x), \forall x \in S$
- *Jak lokalni optimum* - najbolje rešenje u nekoj okolini $N \subseteq S, f(x^*_N) < f(x), \forall x \in N$
- *Slab lokalni optimum* - jedno od najboljih rešenja u okolini $N \subseteq S, f(x^*_N) \leq f(x), \forall x \in N$

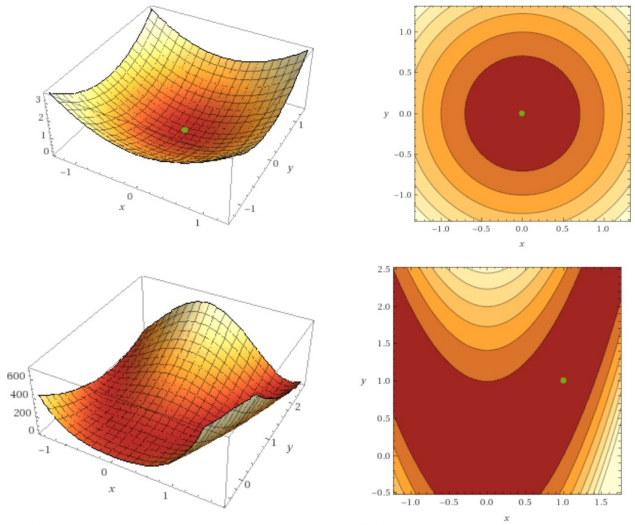
Postoje još neki vidovi postavki optimizacionih problema: *problemi sa višestrukim optimumima* (cilj je pronaći sva rešenja koja su optimalna ili dovoljno blizu optimuma), *višeciljna optimizacija* (problem ima više funkcija cilja pa je složenije urediti rešenja) i *dinamička optimizacija* (funkcija cilja se menja tokom vremena).
Prema tipu domena nad kojim se vrši, optimizacija se deli na: *kombinatornu (diskretnu) optimizaciju* (dopustivi skup vrednosti promenljivih je iz konačnog ili beskonačnog skupa celih brojeva; specijalni slučaj su problemi binarne optimizacije) i *globalnu (kontinualnu) optimizaciju* (dopustivi skup vrednosti iz domena realnih brojeva).

3.2 Optimizacija bez ograničenja, definicija, primer

Formulacija problema je sledeća:

$$minimizovati\ f(x), x = (x_1, x_2, ..., x_{nx}), x \in S$$

Primer: pronaći minimum funkcije $f(x, y) = x^2 + y^2$ na domenu realnih brojeva



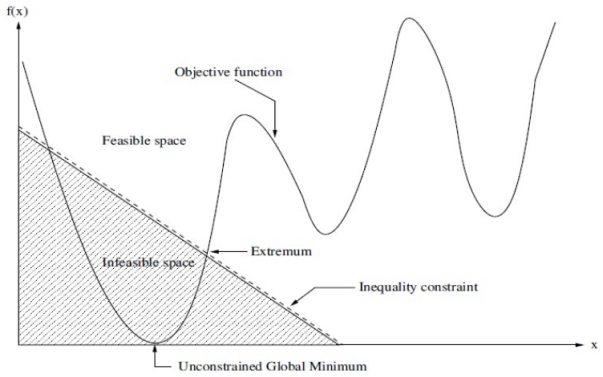
Rosenbrock функција – $f(x,y)=(1-x)^2+100(y-x^2)^2$

3.3 Optimizacija sa ograničenjima, definicija, slika sa objašnjenjima ključnih pojmova, rad sa nedopustivim rešenjima

Opšta forma u slučaju ograničenja:

$$minimizovati\ f(x), x = (x_1, ..., x_{nx}), x \in S$$

pri ograničenjima: $g_m(x) \leq 0, m = 1, ..., n_g$; $h_m(x) = 0, m = n_g + 1, ..., n_g + n_h$. Vidimo da pored domenskih ograničenja ovde postoje i ograničenja zasnovana na jednakosti i/ili nejednakosti. Forma funkcija g_m i h_m je u opštem slučaju nelinearna.



Postavlja se pitanje, šta raditi sa nedopustivim rešenjima?

1. *Odbacivati ih*
2. *Dodeljivati penal* – npr. negativni faktor u slučaju maksimizacije ili pozitivan u slučaju minimizacije
3. *Svoditi na rešenje bez ograničenja* pa ga posle konvertovati u rešenje koje poštuje ograničenja
4. *Održavati dopustivost* samim dizajnom metoda
5. *Uređivati nedopustiva* rešenja prema stepenu nedopustivosti
6. *Popravljati* nedopustiva rešenja.

3.4 Kombinatorna optimizacija i optimizacioni algoritmi

Formulacija problema kombinatorne optimizacije je:

$$minimizovati\ f(x), x = (x_1, x_2, ..., x_{nx}), x \in S$$

pri čemu je S konačan ili beskonačan i diskretan.

Primer: Trgovački putnik (TSP) - neka je zadat skup C od m gradova i funkcija udaljenosti $d(c_i, c_j) \in N$ za svaki par gradova. Pronaći permutaciju $p : [1..m] \rightarrow [1..m]$ takvu da je ukupna suma udaljenosti grana koje se prolaze obilaskom minimalna.
Kolika je veličina dopustivog skupa? Veličina je m!. Pomoću Stirlingove formule može se uočiti da je ovo eksponencijalno.
Koji algoritam preporučujete u opštem slučaju i šta je opšti slučaj? Opšti slučaj bi bio TSP nad proizvoljnim grafom. U opštem slučaju, najsigurnije rešenje je totalna enumeracija (eng. Brute-force).
Šta ako pretpostavimo da se radi u Euklidskom prostoru? Može se malo bolje zbog važenja nejednakosti trougla, ali je problem i dalje težak.
Poređenje različitih pristupa:
Prva dva pristupa (brute force i dinamičko programiranje) su neadekvatni kada dimenzija problema naraste – npr. već za 50-ak gradova. Međutim, prednost je što daju optimalna rešenja ako završe.
Aproksimativni pristup je efikasan –polinomijalan. Međutim, kvalitet rešenja može biti značajno narušen, ponekad nam dobijanje do dva puta lošijeg rešenja ne odgovara. Prednost je što znamo da rešenje ne može da bude više od 2 puta lošije.

3.4.1 Algoritmi pretrage

Metaheuristike pripadaju široj grupi algoritama pretrage. Opšta forma algoritma lokalne pretrage:

```
Zapocni od polaznog resenja x(0) iz S;
t = 0;
repeat
    Izracunaj vrednost f(x(t));
    Izracunaj pravac i smer pretrage, q(t);
    Izracunaj duzinu koraka pretrage n(t);
    Predji u naredno resenje x(t+1) -> x(t) + n(t)q(t);
    t = t + 1;
until nije zadovoljen kriterijum zavrsetka;
Vrati x(t) kao konacno resenje;
```

Dobri algoritmi pretrage koji ne „upadaju“ u lokalne optimume koriste i slučajnosti determinizam (simulirano kaljenje):

```
Zapocni od polaznog resenja, x(0);
Postavi pocetnu temperaturu, T(0);
t = 0;
repeat
    Formiraj novo resenje, x;
    Izracunaj vrednost, f(x);
    Izracunaj verovatnocu prihvatanja resenja;
    if U(0, 1) <= verovatnoca prihvatanja then
        x(t) = x;
    end
until nije zadovoljen kriterijum zavrsetka;
Vrati x(t) kao konacno resenje;
```

3.5 Višeciljna optimizacija

Često u praksi situacija da je potrebno zadovoljiti više funkcija cilja (kriterijuma): ekonomija – naći portfolio akcija sa maksimalnim prihodom i minimalnim rizikom; transportni problemi – maksimizacija iskorišćenosti ulica uz minimizaciju zagušenja, troškova rutiranja i slično. Definicija *optimalnosti* je komplikovanija. Često poboljšanje jedne funkcije cilja izaziva pogoršanje druge, npr. poboljšanje strukturalne stabilnosti mosta izaziva pogoršanje (povećavanje) troškova izgradnje. Tada se govori o pravljenju balansa, odnosno kompromisa koncept *nedominiranih rešenja*.
Pristupi rešavanju:

- *Pravljenje ponderisanih proseka odnosno agregacija.* Za svaku funkciju cilja se određuje težina (značaj). Ovo se kasnije svodi na klasičnu jednociljnu optimizaciju. Problem: kako odrediti pondere/težine?

- *Pravljenje skupa Pareto-optimalnih rešenja.* Rešenje x dominira nad rešenjem y ako nijedna vrednost funkcije cilja od rešenja y nije bolja od odgovarajuće vrednosti funkcije cilja x. Rešenje x je Pareto-optimalno ako ne postoji nijedno drugo rešenje koje dominira nad njim. Skup svih Pareto-optimalnih rešenja se naziva Pareto-optimalan skup. Pareto-optimalna površ predstalja površ koju formiraju funkcije cilja kada se primene nad Pareto-optimalnim skupom rešenja. Pristupi rešavanju: algoritmi pretrage koji mogu efikasno da pretražuju Pareto-optimalnu površ (npr. populacione strategije koje iterativno poboljšavaju skup dobrih rešenja upotrebom svojstva dominacije).

3.6 Klase složenosti izračunavanja i rešavanje NP teških problema

Algoritam za rešavanje nekog problema je: konačan spisak pravila čijim praćenjem dolazimo do rešenja bilo kojeg partikularnog problema (instance problema) iz zadate klase. Praćenje pravila traje konačno mnogo koraka.
Problemi odlučivanja: rešenje se sastoji u potvrđivanju ili opovrgavanju neke osobine.
Svođenje na problem odlučivanja: različiti optimizacioni problemi poput TSP se mogu svesti na problem odlučivanja. Možemo se npr. pitati da li za konkretan TSP problem postoji rešenje sa troškom manjim od C. Dalje, možemo na osnovu odgovara menjati granicu troška i tako iterativno.
Problem odlučivanja pripada ***klasi P*** (polinomsko rešivih) ako postoji algoritam A za rešavanje tog problema i polinom $p(n)$ takav da A završava izvršavanje za ne više od $p(n)$ koraka za svaku instancu tog problema, pri čemu je n dimenzija problema. Polinomijalne

algoritme smatramo „dobrim“ algoritmima. Algoritme čije vreme ne možemo da ograničimo polinomom podrazumevano ograničavamo eksponencijalnom funkcijom c^n (c>1).
Postoje problemi za koje je dokazano da ne mogu biti rešeni algoritmom bržim od ***eksponencijalnog***. Podrazumeva se da su ovi problemi postavljeni kao problemi odlučivanja (klasa EXPTIME). Npr. problemi evaluacije poteza u uopštenom šahu, igri GO i slično. Uopteśna varijanta igre podrazumeva da je promenljive dimenzije. Primer: pronaći skup svih razapinjućih stabala u potpunom (kompletnom) grafu sa n čvorova.
Međutim, postoje problemi za koje se ne zna da li postoji polinomski algoritam za njihovo rešavanje. Ako se za konkretno ponuđeno rešenje problema odlučivanja može utvrditi da li je odgovor potvrdan u polinomijalnom broju koraka, onda je u pitanju neterministički polinomski problem (***NP problem***). Na primeru TSP se može primetiti da se za: bilo koju datu permutaciju gradova (rešenje) i broj C koji predstavlja trošak može dati potvrdan ili odričan odgovor u polinomskom vremenu.

Neka su data dva problema odlučivanja A_1 i A_2. Pretpostavimo da se za A_1 može konstruisati polinomski algoritam u kojem se kao jedan od koraka pojavljuje algoritam za rešavanje problema A_2. Ako je pritom algoritam za rešavanje problema A_2 polinomski onda je jasno da i za A_1 postoji polinomski algoritam. Kaže se da se A_1 redukuje na A_2 - ako se za svaki specijalan slučaj X problema A_1 može u polinomskom vremenu pronaći specijalan slučaj Y problema A_2 takav da je za problem X odgovor potvrdan akko je i za Y odgovor potvrdan.

Problem je ***NP potpun*** ako za svođenje bilo kog NP problema na posmatrani problem postoji polinomski algoritam. Posledica: ako se za bilo koji NP potpun problem pronađe polinomski algoritam, time se dokazuje postojanje polinomskog algoritma za svaki NP problem! Odnosno pokazalo bi se da je N=PN! U kontekstu optimizacije spominju se NP teški problemi. To su problemi čije su odlučive varijante NP potpuni problemi.
Kako rešavati probleme? Dva opšta pristupa:

1. *Egzaktno rešavanje problema* - postupci koji dovode do garantovano optimalnog rešenja ako završe

2. *Približno (aproksimativno) rešavanje* - postpuci koji čak i kada završe ne garantuju optimalnost. Ovde ubrajamo i (meta)heuristike i algoritme sa garancijom kvaliteta

Polinomske probleme rešavati egzaktno! Približno rešavati samo ako ne postoji polinomski postupak, a ovo znači da približno treba rešavati NP teške probleme!

4 Evolutivna izračunavanja

4.1 Evolutivna izračunavanja - opšti koncepti

Evolucija se može smatrati optimizacionim procesom sa ciljem poboljšanja prilagođenosti organizma (ili sistema) dinamičnom i takmičarski nastrojenom okruženju.
Domeni: hemijski, kosmički, biološki, ljudske tvorevine...

Prirodna selekcija - svaka jedinka se takmiči sa ostalima u cilju preživljavanja. „Najbolje“ jedinke imaju veću šansu da prežive i ostave potomstvo. Zbog ovoga češće prenose svoje gene, tj. karakteristike. Tokom vremena, ovakve „pogodne“ karakteristike postaju dominantne u populaciji. Tokom stvaranja potomaka ulogu igraju i slučajni događaji: kroz ukrštanje (bira se gen oca ili majke) ili kroz procese mutacije (nasumične izmene zbog spoljnih događaja). Neki slučajni događaji mogu dodatno da unaprede organizam.

Evolutivna izračunavanja - imitiraju proces evolucije kroz: prirodnu selekciju, ukrštanje, mutaciju itd. Umesto organizama i njihove borbe za preživljavanjem, jedinke u populacijama, kodiraju rešenja nekog problema. Nakon nekog vremena, rešenja evoluiraju u smeru poboljšanja.
Evolutivni algoritmi traže optimalna rešenja putem stohastičke pretrage nad prostorom rešenja. Jedinke (hromozomi) predstavljaju pojedinačne tačke u prostoru rešenja. Ključni aspekti EA su:

1. Kodiranje rešenja u vidu hromozoma – npr. niz celih brojeva
2. Fitnes funkcija – ocena kvaliteta jedinke
3. Inicijalizacija početnog skupa jedinki (početnih rešenja)
4. Operatori selekcije – kako biramo one koji se reprodukuju
5. Operatori ukrštanja – kako se vrši stvaranje novih jedinki

```
Inicijalizuj broj generacija na t = 0;
Kreiraj i inicijalizuj nx–dimenzionu populaciju C(0) od ns jedinki;
while nisu zadovoljeni uslovi za zavrsetak do
    Izracunaj fitnes f(xi(t)), svake jedinke xi(t);
    Izvrsi ukrstanje i formiraj potomke;
    Odaberi novu populaciju, C(t + 1);
    Predji u narednu generaciju, t = t + 1;
end
```

Evolutivni algoritmi:

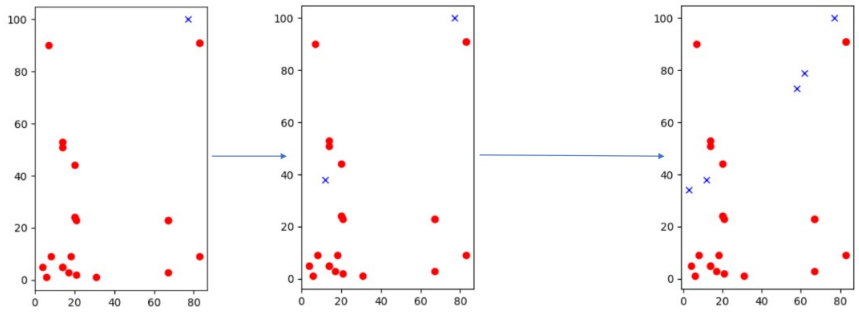
- *Genetski algoritmi* - evolucija nad linearnim genotipom, nizom
- *Genetsko programiranje* - evolucija nad stabloidnim genotipom
- *Evolutivno programiranje* - evolucija fenotipa tj. ponašanja. Kodiranje sekvencama ponašanja, a ne nizovima. Nema ukrštanja. Fitnes je relativan u odnosu na druge jedinke.
- *Evolutivne strategije* - evolucija evolucije: evolucija genotipa + evolucija paramatera evolucije genotipa
- *Diferencijalna evolucija* - kao standardni EA samo se mutacija bira iz unapred nepoznate slučajne raspodele – prilagođene populaciji. Biraju se vektori pomeraja koji su relativni u odnosu na ostale jedinke populacije.

- *Kulturna evolucija* - evolucija kulture u populacijama – jedinke prihvataju verovanja iz populacije, ali i utiču na populaciju srazmerno svojoj prilagođenosti.
- *Koevolucija* - evolucija i preživljavanje jedinki kroz saradnju i takmičenje, npr. biljke i insekti (simbioza)

4.2 Kodiranje rešenja evolutivnog algoritma, fitnes funkcija i inicijalna populacija

Kodiranje (reprezentacija) - *hromozom*. Hromozomi sačinjeni od molekula DNA (nalaze se u jezgru ćelije). Svaki hromozom sačinjen od velikog broja gena. Gen je jedinica nasleđivanja. Određuje anatomiju i fiziologiju organizma jer kodira i kontroliše proces izgradnje proteina. Jedinka je sačinjena od sekvence gena. Vrednost (sadržaj) gena se zove genski alel. U kontekstu EA – hromozom predstavlja *rešenje problema* dok su pojedinačni geni *karakteristike* rešenja.
Odabir pogodnog kodiranja je ključno za rešavanje problema. Kodiranje je najčešće zasnovano na nizu vrednosti nekog tipa, osim u slučaju genetskog programiranja gde je kod nelinearan (stablo). Klasičan primer reprezentacije: binarni vektor fiksne dužine. Kodiranje može biti zasnovano i na nizu celih brojeva fiksne dužine.
Ponekad se domen hromozoma i domen rešenja ne poklapaju. Na primer, možemo koristiti niz realnih vrednosti za hromozom, a da rešenje bude binarne prirode.

Primer:
p-Median problem je definisan nad grafom, a njegovo rešenje je predstavljeno podskupom od p odabranih čvorova takvih da je ukupna udaljenost do svih ostalih čvorova minimalna. Problem koji ćemo razmotriti je definisan slično s tim što nije dat graf već ravan tj. pravougaonik određenih dimenzija. Umesto udaljenosti izražene u broju grana duž puta ovde je udaljenost jednostavno euklidska. Razlog je to što želimo da omogućimo laku vizuelizaciju rešenja.
Kodiranje za 5-Median u ravni - najjednostavnija reprezentacija je vektora realnih vrednosti (neke metode ne podržavaju realnu reprezentaciju).



Prema Darvinomo modelu evolucije, jedinke sa najboljim karakteristikama imaju šanse da prežive i ostave potomstvo. Kvantifikacija ovih karakteristika se izražava putem tzv. ***fitnes funkcije***. Fitnes funkcija se primenjuje nad jedinkom. Ova vrednost je obično apsolutna mera kvaliteta jedinke. Međutim, nekada može biti i relativna u odnosu na druge jedinke. Fitnes funkcija je obično, ali ne i nužno, jednaka funkciji cilja. U prethodnom primeru, za 5-Median u ravni fitnes funkciju definišemo na isti način kao i funkciju cilja, a to je ukupnu udaljenost svih tačaka do najbliže odabrane tačke.

Za validaciju kvaliteta predloženog algoritma dobro je imati uporedni algoritam. Idealno je ako je taj algoritam egzaktan, tj. radi tačno. U slučaju NP teških problema dimenzija koju rešavamo uporednim algoritmom je očekivano mala. Drugi način je poređenje sa već postojećim rezultatima iz literature. Često uporedni algoritmi koriste istu funkciju cilja i kodiranje.
Inicijalizacija rešenja - standardni pristup: inicijalnu populaciju formirati od nasumično odabranih dopustivih rešenj, u slučaju nedopustivih rešenja verovatno će biti potrebna popravka. Zašto? Zbog boljeg pokrivanja celog skupa dopustivih rešenja. Dovoljno velik slučajan uzorak ima dobru reprezentativnost. U slučaju da neki deo prostora rešenja nije pokriven u početku velika je šansa da se neće kasnije uopšte obići taj deo.
Kolika treba da bude veličina populacije i koje su prednosti/mane velike/male populacije? Veličina se određuje obično empirijski za konkretnu metodu. Velika populacija omogućava veću „pokrivenost“ i povećava šansu za nalaženjem globalnog optimuma (diverzifikacija). Mala populacija je efikasnija i omogućava bržu konvergenciju ka lokalnom optimumu (intenzifikacija).

4.3 Operator selekcije kod evolutivnog algoritma i elitizam

Selekcija - u ovoj fazi se vrši odabir rešenja koja treba da ostave potomstvo. Načelna ideja: „daj veću šansu boljim rešenjima“. *Selekcioni pritisak* (eng. selection pressure) - vreme potrebno da se proizvede uniformna populacija jedinki odnosno da najbolje jedinke ostave svoje gene svuda. Operatori selekcije sa visokim selekcionim pritiskom smanjuju raznovrsnost gena u populaciji brže (preuranjena konvergencija). Pristupi:

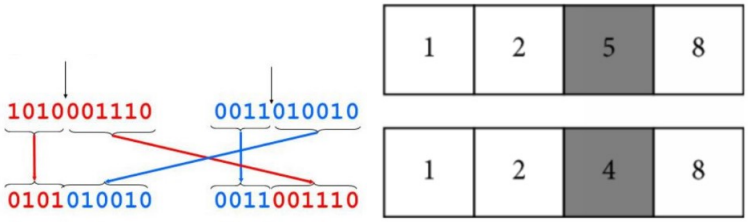
- *Slučajna* – svaka jedinka ima istu šansu. Najniži selekcioni pritisak, ali spora konvergencija
- *Proporcionalna* – daje veću šansu boljim jedinkama. Ruletska selekcija je standarni način implementacije ovog mehanizma
- *Turnirska selekcija* – turnir između slučajnog podskupa jedinki. Ako je podskup jednak populaciji – strategija je elitistička. Ako je podskup veličine jedan – strategija je slučajna. Variranjem veličine podskupa menja se selekcioni pritisak
- *Rangovska selekcija* –umesto vrednosti fitnes funkcije koristi se samo redni broj u uređenju populacije. Smanjuje selekcioni pritisak, jer se jako dobrim rešenjima relativizuje značaj

Elitizam - tehnika koja sprečava „gubljenje“ dobrih jedinki. Iako se najbolje jedinke najverovatnije nalaze u skupu odabranih roditelja, primenom ukrštanja i mutacije može doći do toga da potomci imaju lošiji kvalitet. Pristup: izaberi najbolju ili nekoliko najboljih jedinki i direktno ih prekopiraj u narednu generaciju; broj elitistički odabranih jedinki ne sme biti preveliki! (visok selekcioni pritisak).

4.4 Operator ukrštanja, mutacije i kriterijum zaustavljana - ukratko

Ukrštanje - ovim procesom se kreiraju nove jedinke – potomci. Podrazumeva upotrebu sledećih operatora:

- *Operatori ukrštanja* – rekombinacije gena
- *Operatori slučajne mutacije* – promene nasumičnih gena

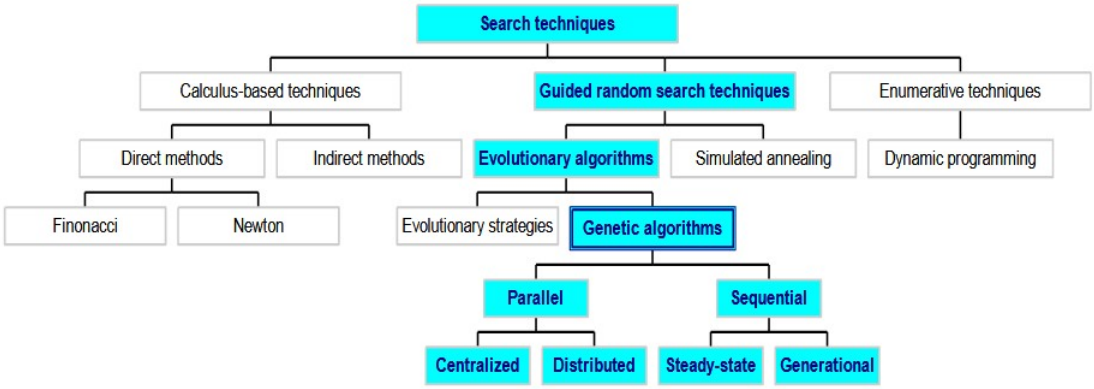


Kriterijumi zaustavljanja:

1. Nakon isteka unapred fiksiranog broja generacija
2. Nakon isteka unapred fiksiranog vremena
3. Kada nema unapređenja u P poslednjih generacija
4. Kada u P poslednjih generacija nema promene u genotipu
5. Kada je nađeno prihvatljivo rešenje: samo ako znamo šta nam je prihvatljivo
6. Kada se nagib fitnes funkcije više ne povećava: potrebno je pratiti kretanje fitnes funkcije kroz vreme

5 Genetski algoritmi

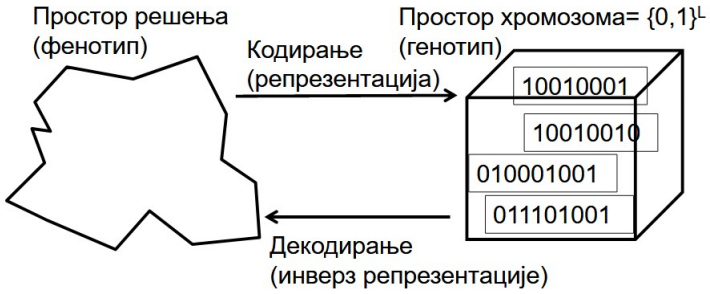
5.1 Uvodni koncepti. Kanonski genetski algoritam



Genetski algoritam (GA) - razvijen u Americi 1970-ih. Ključni autori: J. Holland, K. DeJong, D. Goldberg. Glavne primene - problemi u diskretnom domenu. Karakteristike: nije preterano brz – kao i većina populacionih metaheuristika; dobra heuristika za rešavanje kombinatornih problema; dosta varijanti, npr. različiti mehanizmi ukrštanja, mutacije, itd.
Jednostavni genetski algoritam (SGA) - originalni genetski algoritam (GA) koji je razvio John Holland se naziva još i jednostavni (kanonski) GA ili SGA (eng. Simple GA). Drugi GA se razlikuju u: reprezentacijama (kodiranjima i dekodiranjima), mutacijama, ukrštanju, selekciji.

```
Inicijalizuj populaciju;
Evaluiraj populaciju; // izracunavanje fitnesa hromozoma
while Nije zadovoljen uslov za zavrsetak {
    Odaberi roditelje za ukrstanje;
    Izvrsi ukrstanje i mutaciju;
    Evaluiraj populaciju;
}
```

Карактеристика GA	Имплементација у оквиру SGA
Репрезентација	Низ битова
Укрштање	n-позиционо или равномерно укрштање
Мутација	Извртање битова са фиксном вероватноћом
Селекција родитеља	Фитнес-сразмерна
Селекција преживелих	Родитељи се потпуно замењују децом
Специјалност	Фокус је на укрштању



SGA Ukrštanje:

1. Odaberi roditelje u skup za ukrštanje (veličina skupa za ukrštanje = veličina populacije)
2. Razbacaj (eng. Shuffle) skup za ukrštanje
3. Za svaki par uzastopnih hromozoma primenjuje se ukrštanje sa verovatnoćom p_c, a ako se ne primeni, onda se kopiraju roditelji
4. Za svako dete primenjuje se mutacija sa verovatnoćom p_m po svakom bitu nezavisno
5. Zameni celu populaciju sa novodobijenom populacijom dece

SGA operator ukrštanja sa jednom tačkom - odaberi slučajnu poziciju (manju od broja gena). Razdvoji svakog roditelja po ovoj poziciji na dva dela. Kreiraj decu razmenom delova između roditelja. p_c je obično iz intervala [0.6, 0.9].
SGA operator mutacije - svaki gen (bit) sa verovatnoćom p_m. p_m se naziva stopa mutacije. Tipično ima vrednost između (1/veličina populacije) i (1/dužina hromozoma).
SGA operator selekcije - osnovna ideja: bolje jedinke imaju veću šansu. Šanse su srazmerne fitnesu. Implementacija - ruletski točak (dodeli svakoj jedinki isečak točka; okreni točak n puta za odabir n jedinki).
SGA je i dalje tema mnogih studija - relevantan metod za poređenje (eng. benchmark) sa drugim GA. Mnoga ograničenja: reprezentacija je previše restriktivna; mutacija i ukrštanje primenljivi samo za bitovsku ili celobrojnu reprezentaciju; selekcija osetljiva na slučaj kada populacija konvergira (fitnes vrednosti bliske); generisanje populacije se može unaprediti tehnikom eksplicitnog preživljavanja.

5.2 Ostali tipovi reprezentacija kod genetskih algoritama i mutacije nad njima

Grejovo kodiranje celih brojeva (idalje binarni hromozomi) - ponekad je pogodnije, jer malim promenama u genotipu se prave i male promene u fenotiup (za razliku od standardnog binarnog koda). "Glatkije" genotip-fenotip preslikavanje može da poboljša rad GA. Danas je, međutim, opšte prihvaćeno da je bolje kodirati numeričke vrednosti direktno kao: cele brojeve, realne brojeve u fiksnom zarezu. Ovo zahteva da i operatori ne budu dizajnirani da rade sa binarnim brojevima, već sa odgovarajućim tipom celim/realnim brojevima.
Direktna celobrojna reprezentacija - neki problemi prirodno imaju celobrojnu reprezentaciju rešenja, npr. vrednosti parametara u procesiranju slika. Neki drugi mogu imati kategoričke vrednosti iz fiksiranog skupa, npr. plavo, zeleno, žuto, roze. *n-poziciono/ravnomerno ukrštanje* radi u ovim situacijama. Binarna mutacija se mora proširiti (ne može biti samo izvrtanje bitova) - mutiranje u bliske (slične) vrednosti; mutiranje u nasumične vrednosti

(tipično kod kategoričnih promenljivih).
Šta ako problem ima rešenje sa realnom reprezentacijom, npr. u problemima globalne optimizacije $f : R^n \to R$. Tipičan test primer: Ackley-jeva funkcija.
Preslikavanje realnih vrednosti na nizove bitova $z \in [x, y] \in R$ predstavljeni kao niz bitova $\{a_1, ..., a_L\} \in \{0, 1\}^L$. $[x, y] \to \{0, 1\}^L$ mora biti inverzno (jedan fenotip za svaki genotip). $\Gamma : \{0, 1\}^L \to [x, y]$ definiše reprezentaciju: $\Gamma(a_1, ..., a_L) = x + \frac{y-x}{2^L-1}(\sum_{j=0}^{L-1} a_{L-j}2^j) \in [x, y]$. Samo 2^L vrednosti od mogućih beskonačno je moguće kodirati. L determiniše preciznost rešenja. Velika preciznost $\to$ dugački hromozomi (spora evolucija). Alternativno, kodiranje može biti direktno uz doradu operatora.
Mutacija za direktno realno kodiranje. Opšta šema za brojeve u fiksnom zarezu:

$$x =< x_1, ...x_l > \to x' =< x'_1, ..., x'_l >, x_i, x'_i \in [LB_i, UB_i]$$

Ravnomerna mutacija: x'_i se bira ravnomerno iz $[LB_i, UB_i]$. Analogno izvrtanju bitova (binarni kod) ili nasumičnom mutiranju (kod celih brojeva). *Neravnomerne mutacije* - postoje mutacije čija se verovatnoća menja sa vremenom, pozicijom, itd. Standardni pristup je dodeljivanje slučajne devijacije svakoj promenljivoj, a potom izvlačenje promenljivih is $N(0, \sigma)$. Standardna devijacija σ kontroliše udeo promena (2/3 devijacija će se nalaziti u opsegu $(-\sigma to + \sigma)$.

Problemi zasnovani na permutacijama. Postoje mnogi problemi koji za rešenje imaju uređenu strukturu: linearnu, kvadratnu, hijerarhijsku, itd. Zadatak je organizovati objekte u odgovarajućem redosledu (npr. problem sortiranja ili problem trgovačnog putnika (TSP)). Ovakvi problemi se generalno izražavaju posredstvom permutacija: ako postoji n promenljivih, onda je reprezentacija sačinjena od n celih brojeva, takvih da se svaki pojavljuje tačno jednom.
Primer TSP
Problem: Neka je dato n gradova. Pronaći rutu sa minimalnom dužinom.
Kodiranje: Označi gradove sa 1, 2,..., n. Jedna kompletna ruta je jedna permutacija (npr. za $n = 4$ [1,2,3,4], [3,4,2,1] su u redu).
Prostor pretrage je VELIK: za 30 gradova, $30! = 10^{32}$ mogućih ruta.

Mutacija nad permutacijama. Normalni operatori mutacije nad permutacijama dovode do nedopustivih rešenja. Na primer operator koji gen i sa vrednošću j menja vrednost u neku vrednost k bi mogao da dovede do toga da se vrednost k pojavljuje više puta u rešenju, dok vrednosti j više nema uopšte u rešenju. Zbog toga se moraju menjati vrednosti bar dvema promenljivama. Verovatnoća mutacije sada opisuje verovatnoću primene operatora nad celim rešenjem, a ne nad pojedinačnim pozicijama.
Mutacija zasnovana na umetanju. Izaberu se dve vrednosti (dva alela) na slučajan način. Drugi se umeće tako da bude posle prvog, pri čemu se svi ostali pomeraju ukoliko je potrebno. Ovo zadržava veći deo uređenja odnosno informacije o prethodnom susedstvu. To je dobro, jer mutacija ne treba da izaziva dramatične promene.
Mutacija zasnovana na zameni. Izaberu se dve vrednosti na slučajan način i zamene se. Zadržava većinu uređenja.
Mutacija zasnovana na inverziji. Izaberu se dve vrednosti na slučajan način, a potom se obrne redosled vrednosti između njih. Ovo je malo intenzivnija promena uređenja od prethodna dva pristupa.
Mutacija zasnovana na mešanju. Izabere se podskup pozicija na slučajan način. Slučajno se reorganizuju vrednosti na tim pozicijama.

5.3 Ostali operatori ukrštanja kod genetskih algoritama

Kvalitet *jednopozicionog* ukrštanja zavisi od redosleda promenljivih u reprezentaciji rešenja. Veća je šansa da će geni koji su blizu biti zadržani u potomstvu. Takođe, geni koji su na različitim krajevima hromozoma se ne mogu naći u istom potomku. Ovo se zove pozициona pristrasnost. Može biti korisna ukoliko znamo strukturu problema, međutim, u opštem slučaju je nepoželjna.
n-poziciono ukrštanje - bira se n slučajnih pozicija. Razdvaja se po tim pozicijama. Spajaju se alternirajući delovi. Ovo je uopštenje jednopozicionog ukrštanja, u kojem i dalje postoji poziciona pristrasnost.
Ravnomerno ukrštanje - kao kod bacanja novčića, dodeljuju se 'glave' jednom roditelju, 'pismo' drugom. Baca se novčić za svaki gen prvog deteta i uzima gen iz odgovarajućeg roditelja. Drugo dete je inverz prvog. Nasleđivanje je stoga nezavisno od pozicije.

Ukrštanje ili mutacija? Debata duga nekoliko decenija. Odgovor: Zavisi od problema, ali najbolje je da postoje oba pošto imaju različite uloge. Samo mutacijski EA su mogući, dok samo ukrštajući EA ne bi radili. Eksploracija - otkrivanje novih oblasti u prostoru pretrage. Eksploatacija - optimizacija u okviru postojećih oblasti (kombinovanje rešenja). Postoji kooperacija i konkurencija između njih. Ukrštanje radi *eksploataciju*, pravi kombinacije „između“ roditeljskih hromozoma (ako neki alel potreban za globalni optimum ne postoji, onda globalno rešenje nikad neće biti dostignuto). Mutacija je dominantno *eksplorativna*, pošto uvodi novu informaciju i time proširuje prostor pretrage (mutacija vrši i eksploataciju, jer menja lokalnu okolinu trenutnog rešenja). Samo ukrštanje može da kombinuje informacije dva roditelja. Samo mutacija može da uvede nove informacije (genski aleli). Ukrštanje ne menja frekvenciju genskih alela u okviru populacije. Da bi se pogodio optimum, obično je potrebna „srećna“ mutacija.

Ukrštanje za direktno realno kodiranje. Kod diskretnog domena (binarni ili celobrojni) svaki alel deteta z je direktno nasleđen od nekog od roditelja (x, y) sa jednakom verovatnoćom: $z_i = x_i$ *or* y_i. Ovde nema smisla koristiti n-poziciono ili ravnomerno. Skica rešenja: formiranje dece koji su "između" roditelja (tzv. aritmetičko ukrštanje), $z_i = \alpha x_i + (1 - \alpha)y_i$ gde je $\alpha : 0 \leq \alpha \geq 1$. Parametar α može biti: konstanta (ravnomerno aritmetičko ukrštanje), promenljiva (npr. zavisi od starosti populacije), odabran slučajno svaki put.
Jednostruko aritmetičko ukrštanje. Roditelji: $< x_1, ..., x_n >$ *and* $< y_1, ..., y_n >$. Odaberi jedan gen (k) slučajno.
Jednostavno aritmetičko ukrštanje. Roditelji: $< x_1, ..., x_n >$ *and* $< y_1, ..., y_n >$. Odaberi slučajni gen *(k)* koji određuje poziciju.
Celovito aritmetičko ukrštanje.. Najčešće korišćeno (zadržavanje 1 deteta – duplo više ukrštanja). Roditelji: $< x_1, ..., x_n >$ *and* $< y_1, ..., y_n >$.

Ukrštanje u permutacijskim problemima. Normalni operatori ukrštanja dovode do nedopustivih rešenja. Nije validno da na primer ukrstimo dve permutacije [123—45] i [543—21] i dobijemo [12321] i [54345]. Predloženi su mnogi specijalizovani operatori koji se fokusiraju na kombinovanje poretka ili susedstva kod dva roditelja.

1. Ukrštanje prvog reda - ideja je da se zadrži relativno uređenje. Opšta šema:

```
Odabrati segment hromozoma prvog roditelja
Iskopirati ovaj segment u prvo dete
Iskopirati preostale vrednosti (brojeve) tako da:
    Kopiranje pocinje desno od kopiranog segmenta
    Koriscenjem redosleda datog drugim roditeljem
Slicno se radi i za drugo dete, ali obrnemo roditelje
```

2. Delimično ukrštanje (PMX). Opšta šema za roditelje P1 i P2:

```
Odabrati slucajni segment i kopirati ga od P1
Pocev od pozicije pocetka segmenta traziti elemente u tom segmentu za P2
    koji nisu bili kopirani
Za svaki od ovih i pronadji vrednost j iz P1 koja je kopirana na njegovo
    mesto
Postavi i na poziciju zauzetu sa j u P2, posto znamo sigurno da j nece
    biti tamo (on je vec u detetu)
Ako je mesto na kojem se nalazi j u P2 vec zauzeto vrednoscu k, onda
    postavi i na poziciju koju zauzima k u P2
Na kraju se preostali elementi samo iskopiraju iz P2
Drugo dete se kreira analogno
```

5.4 Populacioni modeli i selekcija

GA se razlikuju od bioloških modela po tome što su veličine populacija fiksne. Ovo nam omogućava da se proces selekcije opisuje pomoću dve stvari: selekcija roditelja i strategija zamene koja odlučuje da li će potomci zameniti roditelje, i koje roditelje da zamene onda. Dve klase GA se razlikuju u zavisnosti od odabrane strategije: generacijski genetski algoritmi (GGA) i genetski algoritmi sa stabilnim stanjem (Steady State GA, SSGA).

SGA koristi tzv. *Generacijski model* (GGA) - svaka jedinka preživi tačno jednu generaciju. Ceo skup roditelja je zamenjen svojim potomcima nakon što su svi potomci stvoreni i mutirani. Kao rezultat ovoga nema preklapanja između stare i nove populacije (pretpostavljamo da se ne koristi elitizam).
Model sa *Stabilnim stanjem* (SSGA) - jedno dete se generiše po generaciji. Jedan član populacije biva zamenjen njime. Čim se potomak stvori i mutira donosi se odluka o tome da li će roditelj ili potomak preživeti i dospeti u sledeću generaciju. Postoji preklapanje između stare i nove generacije.
Generacijski jaz - količina preklapanja između trenutne i nove generacije. Za GGA je generacijski jaz 1.0, dok je za SSGA 1/veličinaPopulacije.

Takmičenje zasnovano na fitnesu. Selekcija se može javiti u dva navrata – selekcija roditelja za ukrštanje ili selekcija preživelih – biranje iz skupa roditelja + deca onih koji će preći u narednu generaciju. Razlike među selekcijama prave se na osnovu: operatora (definišu različite verovatnoće) i algoritmi (definišu kako su verovatnoće implementirane).
Fitnes srazmerna selekcija. Problem: jedna visoko kvalitetna jedinka može brzo da preuzme čitavu populaciju ako su ostale jedinke značajno lošije – rana konvergencija. Kada su fitnesi slični (pred kraj), selekcioni pritisak je loš (on definiše koliko su favorizovana dobra rešenja). Kada su fitnesi relativno slični (bliski), smanjuje se i favorizacija. Skaliranje može da pomogne u rešavanju prethodna dva problema. Skaliranje prema najgorem: $f'(i) = f(i) – \beta^t$, gde je β najgori fitnes u poslednjih n generacija.
Rang-bazirana selekcija. Pokušaj da se prevaziđu problemi fitnes srazmerne selekcije. Vrednost fitnesa nema apsolutni već relativni značaj ovde. Najbolja jedinka ima najviši rang μ, a najgora rang 1. Trošak primene sortiranja je obično zanemarljiv.
Turnirska selekcija. Prethodne metode se oslanjaju na opšte populacione statistike. Ovo može biti usko grlo, npr. na paralelnim mašinama. Oslanjaju se na prisustvo eksternih fitnes funkcija koje možda ne postoje uvek: npr. evolucija botova za igrice (ovde ne znamo koji je fitnes, ali možemo da utvrdimo ko bolje igra). Opšta šema: odaberi k članova na slučajan način, a potom odaberi najboljeg od njih. Nastavi proces za odabir još jedinki. Verovatnoća odabira jedinke i zavisi od: ranga i, vrednosti (uzorka) k - veće k znači veći selekcioni pritisak, da li se takmičari biraju sa vraćanjem - odabir bez vraćanja pojačava selekcioni pritisak.

Selekcija preživelih. Metode slične onima koje se koriste za odabir roditelja. Dve grupe pristupa postoje kod selekcije preživelih:

- Selekcija zasnovana na starosti - kao kod SGA. SSGA može da implementira brisanje slučajne (potomak menja slučajno izabranu jedinku iz tekuće populacije, ne preporučuje se) ili brisanje najstarije (first in first out, potomak menja najstariju jedinku iz populacije, često se dešava da to bude jedna od najboljih jedinki).
- Fitnes-srazmerna selekcija - primena neke od ranije pomenutih metoda: ruletska, turnirska...

Specijalan slučaj je elitizam. Često se koristi kod oba populaciona modela (GGA, SSGA). Uvek se zadržava kopija najboljeg rešenja do sada.

5.5 Teorema o shemama

Teorijska osnova iza genetskih algoritama i genetskog programiranja (John Holland, sedamdesete godine). Nejednakost koja objašnjava evolutivnu dinamiku.

Teorema (neformalno): Kratke sheme sa natprosečnim fitnesom postaju eksponencijalno učestalije tokom generacija.

Shema je šablon koji identifikuje podskup niski koje su slične na pojedinačnim pozicijama. Primer: za binarne niske dužine 6, primer sheme je 1*10*1 gde ovakva shema opisuje sve niske dužine 6 sa fiksiranim bitovima na 4 opisane pozicije.
Red sheme $o(H)$ je definisan kao broj fiksiranih pozicija. $\sigma(H)$ je udaljenost između prve i poslednje fiksirane pozicije. Fitnes sheme je prosečni fitnes svih niski koje pripadaju shemi.

Teorema:

$$E(m(H,t+1)) \geq \frac{m(H,t)f(H)}{a_t}[1-p]$$

gde je $m(H,t)$ broj niski koje pripadaju shemi H u generaciji t, $f(H)$ prosečni fitnes sheme H, dok je $a(t)$ prosečni fitnes u generaciji t. p je verovatnoća da će ukrštanje ili mutacija „razbiti“ shemu:

$$p = \frac{\delta(H)}{l-1}p_c + o(H)p_m$$

gde je l dužina genotipa dok su p_c i p_m verovatnoće ukrštanja i mutacije.

6 Genetsko programiranje

6.1 Pregled koncepata i opšta shema

Razvijeno u Americi 90-tih godina, J. Koza. Obično se primenjuje u mašinskom učenju (predikcija, klasifikacija...). Konkurentan neuronskim mrežama i sličnim metodama, ali zahteva ogromne populacije (hiljade jedinki). Relativno spor. Specijlane karakteristike: nelinearni hromozomi (stabla, grafovi), mutacija moguća ali nije neophodna.
Smatra se specijalizacijom genetskog algoritma. Slično kao GA, i genetsko programiranje se koncentriše na evoluciju genotipova. Osnovna razlika je u reprezentaciji: tamo gde GA koristi reprezentaciju stringom ili vektorom, GP koristi stablo.

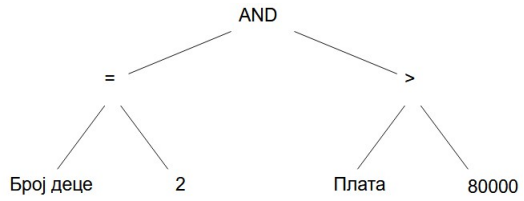
Репрезентација	Стабло
Укрштање	Размена подстабала
Мутација	Случајна промена у дрвету
Селекција родитеља	Фитнес сразмерна
Селекција преживелих	Генерацијска замена

Primer: Određivanje kreditne sposobnosti

Banka hoće da napravi razliku između dobrih i loših kandidata za davanje pozajmica. Potrebno je uzeti u obzir istorijske podatke. Mogući model:

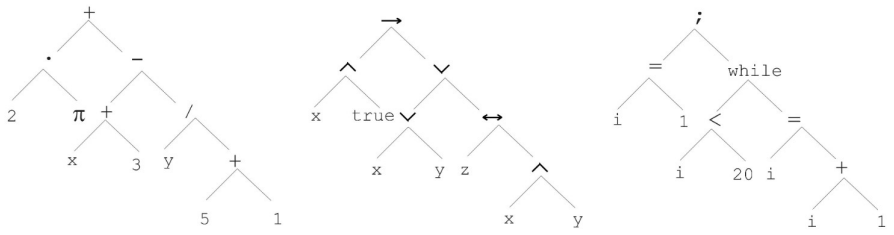
$$\text{IF (broj dece} = 2) \text{ AND (plata} \geq 80000) \text{ THEN dobar ELSE loš}$$

Opšti pristup: *IF formula THEN dobar ELSE loš*. Nepoznata je prava formula. Prostor pretrage (fenotip) je skup svih formula. Fitnes formule: procenat dobro klasifikovanih primera. Prirodna reprezentacija formule (genotip) je stablo. Stablo za gore navedenu formulu:



Reprezentacija stablom - stabla imaju mogućnost predstavljanja velikog broja formula.

- aritmetička formula $2 * \pi + ((x + 3) - \frac{y}{5+1})$
- logička formula $(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$
- program $i = 1; while(i \leq 20)i + +;$

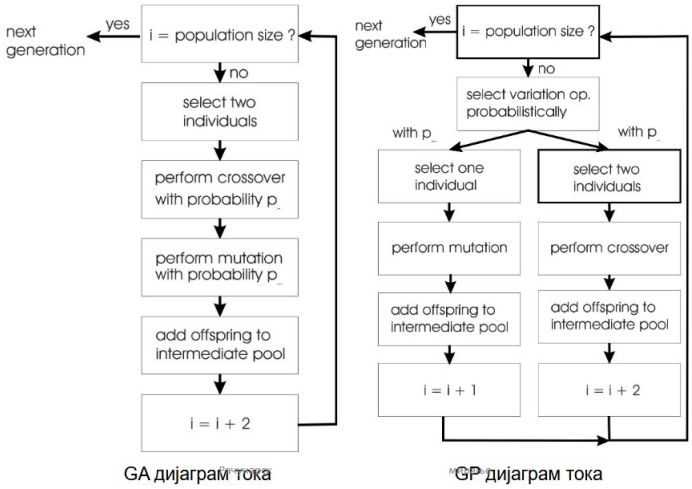


U genetskim algoritmima (GA), evolutivnim strategijama (ES), evolutivnom programiranju (EP), hromozomi su linearne strukture, npr. nizovi bitova, nizovi celih brojeva, nizovi realnih brojeva, permutacije itd. Stablo-hromozomi su nelinearne strukture. Kod GA, ES, EP veličina hromozoma je fiksna, dok stabla u GP mogu da imaju proizvoljnu dubinu i širinu. Simbolički izrazi mogu biti definisani pomoću skupa termova T i skupa funkcija F. Skup termova određuje sve promenljive i konstante, dok skup funkcija sadrži sve funkcije koje se mogu primeniti na elemente skupa termova. Ove funkcije mogu uključivati matematičke, aritmetičke, logičke funkcije, kao i strukture odlučivanja kao if-then-else i petlje. Elementi terminalnog skupa čine listove stabla, dok su svi ostali čvorovi elementi skupa funkcija. Za neki problem prostor pretrage sastoji se od skupa svih mogućih stabala koja se mogu konstruisati korišćenjem definisane gramatike. Kad se definišu ovi skupovi može se koristiti rekurzivna definicija:

1. Svaki $t \in T$ je korektan izraz.
2. $f(e_1, ..., e_n)$ je korektan izraz ako $f \in F$, $arity(f) = n$ i $e_1, ..., e_n$ su korektni izrazi.
3. Ne postoje druge koretkne forme izaza.

U opštem slučaju izrazi u GP nisu tipizirani što znači da svaki $f \in F$ može uzeti bilo koji $g \in F$ kao argument.

Generisanje potomaka. Poređenje: GA koristi ukrštanje *i* mutaciju sekvencijalno (slučajno), a GP koristi ukrštanje *ili* mutaciju (slučajno).



GA дијаграм тока GP дијаграм тока

Selekcija. Selekcija roditelja je obično fitnes-srazmerna. Selekcija u veoma velikim populacijama:

- Rangiraj populaciju prema fitnesu i podeli ih u dve grupe: grupa 1 - najboljih x% populacije i grupa 2 - ostalih (100-x)% populacije.
- 80% operacija selekcije izvrši nad grupom 1, preostalih 20% nad grupom 2.
- Za populacije veličine 1000, 2000, 40000, 8000, x = 32%, 16%, 8%, 4%.
- Ovi procenti su određeni empirijski.

Selekcija preživelih kao standardni pristup ima generacijski. U poslednje vreme postaje popularan model sa stabilnim stanjem i elitizmom.

Inicijalizacija populacije. Početna populacija se generiše na slučajan način unutar ograničenja maksimalne dubine i semantike izražene datom gramatikom. Za svaku jedinku bira se koren nasumično iz funkcijskog skupa. Broj dece korena i svakog čvora koji nije list se određuje pomoću *arity* odabrane funkcije. Za svaki čvor koji nije koreni selektuje se nasumično element iz skupa terminala ili iz funkcijskog skupa. Čim je izabran element iz skupa terminala taj dati čvor postaje list i više se ne razmatra za širenje.
Postavlja se maksimalna dubina stabla D_{max}:

- *Balansirani pristup* (teži se ka balansiranom stablu dužine D_{max}). Čvorovi na dubini $d<D_{max}$ se slučajno biraju iz skupa funkcija F. Čvorovi na dubini $d = D_{max}$ se slučajno biraju iz skupa termova T.
- *Ograničeni pristup* (teži se ka stablu ograničene dubine $\leq D_{max}$). Čvorovi na dubini $d<D_{max}$ se slučajno biraju iz skupa $F \cup T$. Čvorovi na dubini $d = D_{max}$ se slučajno biraju iz skupa T.

Standardna GP inicijalizacija ima kombinovani pristup koji koristi balansirani i ograničeni pristup (svaki po pola populacije).
Pristup zasnovan na povećanju. *Bloat* – tendencija ka 'udebljavanju' - stabla unutar populacije tokom vremena rastu. Potrebne su kontramere, sprečavanje upotrebe operatora koji dovode do 'prevelike' dece, penalizacija 'prevelikih' jedinki.

6.2 Operatori mutacije i ukrštanja kod genetskog programiranja

Mutacija. Najčešći operator mutacije: zameni slučajno odabrano podstablo novim slučajno generisanim stablom. Mutacija ima dva parametra:

- Verovatnoća p_m odabira mutacije, inače se vrši ukrštanje
- Verovatnoća odabira unutrašnje tačke kao korena podstabla

Savetuje se da p_m bude 0 (Koza '92) ili jako blisko 0, npr 0.05 (Banzhaf et al. '98). Veličina deteta može biti veća od veličine roditelja.

Ukrštanje. Najčešći operator ukrštanja: zamena dva slučajno odabrana podstabla između roditelja. Ukrštanje ima dva parametra:

- Verovatnoća p_c za odabir ukrštanja, inače se vrši mutacija
- Verovatnoća odabira unutrašnje tačke (korena podstabla) kao pozicije za ukrštanj kod svakog od roditelja

Veličina deteta može biti veća od veličine roditelja.

7 Inteligencija rojeva - uopšteno

Formalno, roj može biti definisan kao grupa (generalno mobilnih) agenata koji međusobno komuniciraju direktno ili indirektno. Interakcije između agenata dovode do distributivnih kolektivnih strategija za rešavanje problema.
Inteligencija rojeva (Swarm Intelligence, SI) odnosi se na ponašanje koje rešava problem, a proizilazi iz interakcije između agenata, dok se CSI (Computational Swarm Intelligence) odnosi na algoritamske modele takvog ponašanja. SI se može nazivati i inteligencijom kolektiva.
Studije društvenih životinja i insekata dovele su do brojnih računarskih modela inteligencije rojeva. Biološki sistemi rojeva koji su inspirisali ovakve modele uključuju mrave, termite, pčele, paukove, jata ptica, itd. U ovakvim rojevima, kolektivno ponašanje je jako kompleksno. Ono je rezultat šeme interakcija između pojedinaca iz roja tokom vremena. Ono nije vezano samo za pojedinca, i ne može se jednostavno predvideti ili svesti na jednostavno ponašanje pojedinca.
Može doći do pojave dobijanja nekih novih koherentnih struktura, šema i svojstava ili ponašanja u kompleksnom sistemu. Ove strukture, šeme i ponašanja nastaju bez ikakvog sistema koordinacije, već proizilaze iz interakcija pojedinaca u njihovoj lokalnoj, potencijalno promenljivoj životnoj sredini. Ponašanje pojedinca je usko vezano za ponašanje kolektiva: pojedinci oblikuju i diktiraju ponašanje roja. S druge strane, ponašanje roja ima uticaj na uslove pod kojima svaki pojedinac izvršava svoje akcije. Ove akcije mogu promeniti sredinu, a s tim i ponašanja tog pojedinca i njegovih suseda – što opet može promeniti ponašanje čitavog roja. Dakle, najbitniji faktor inteligencije rojeva je *interakcija* ili *saradnja*. Interakcija među pojedincima pomaže u usavršavanju znanja o okolini. Interakcija može biti direktna (fizički kontakt ili vizuelni, zvučni ili hemijski signali) ili indirektna (pomoću lokalnih promena sredine). Termin *stigmergija* se odnosi na indirektnu komunikaciju između pojedinaca.

Primeri iz prirode:

- *Pčele* - koopearcija u okviru kolonije; regulacija temperature unutar saća; efikasnost se postiže specijalizacijom (podela rada u okviru kolonije); komunikacija (izvori hrane se koriste u skladu sa njihovom blizinom saću i kvalitetom).
- *Ose* - tragači za hranom, tragači za vodom, graditelji; složena gnezda; horizonalne kolone; zaštitne opne; centralni ulazni hol.
- *Termiti* - konusni spoljni zidovi i ventilacioni otvori; legla u centralnoj košnici; spiralni ventilacioni otvori za hlađenje; potporni stubovi.
- *Mravi* - prave „autoputeve“ do mesta sa hranom tako što ostavljaju trag feromona; formiraju lance svojim telima u cilju pravljenja mosta preko lišća i slično; podela posla između više i manje bitnih mrava.

Cilj kompjuterskih modela inteligencije rojeva je modeliranje jednostavnog ponašanja pojedinaca, njihove lokalne interakcije sa sredinom i susednim pojedincima, kako bi se dobila kompleksna ponašanja koja se mogu iskoristiit za rešavanje kompleksnih problema. Na primer, PSO (Particle Swarm Optimization) modeluje dve vrste ponašanja:

1. Svaki pojedinac se kreće prema svom najbližem najboljem susedu, i vraća se u okolinu koja se pokazala kao najbolja po njega. Kao rezultat, svi pojedinci će se približavati stanju sredine koje je najbolje za sve pojedince.

2. Optimizacija koja modeluje jednostavno ponašanje u koloniji mrava kada se prate tragovi feromona, gde svaki mrav prepoznaje koncentraciju feromona u svojoj lokalnoj sredini i reaguje tako što bira put sa najvećom koncentracijom. Ovim se bira najbolja varijanta (najkraći put) iz skupa svih varijanti (puteva).

Socijalni insekti - karakteristike:

- Fleksibilnost - kolonija savladava unutrašnje preturbacije, kao i spoljne izazove
- Robusnost - zadaci se završavaju iako neke individue zakažu
- Decentralizovanost - ne postoji centralni mehanizam kontrole niti koncept lidera
- Samoorganizovanost - putevi do rešenja vremenom iskrsnu, nisu unapred predefinisani

Ikosistem simulacija. Cilj simulacije je predstavljanje populacije u kojoj se svi članovi vode istim pravilima. Svaki agent X ima dodeljena dva nasumična protivnika A i B. Pravilo za X je da se postavlja tako da B bude na putu između A i X. Pravilo za X je zatim promenjeno tako da se on sada postavlja između A i B. Mala promena drastično utiče na kolektivno ponašanje.

Problemi sa inteligentnim rojevima - ovakvi sistemi se teško programiraju, jer je probleme koje rešavamo teško prebaciti na jezik inteligentnih rojeva. Rešenja iskrsnu unutar sistema. Rešenja su rezultat ponašanja i interakcija između pojedinačnih agenata (jedinki) unutar sitema.
Glavni sastojci za samoorganizaciju:

- Pozitivna povratna sprega (eng. Positive Feedback)
- Negativna povratna sprega (eng. Negative Feedback)
- Pojačavanje i smanjivanje slučajnosti
- Oslanjanje na međusobne interakcije agenata

Svojstva samoorganizacije:

- Kreiranje struktura (gnezda, tragovi, socijalno uređenje (hijerarhija))
- Promene su rezultat postojanja višestrukih puteva razvoja (nekoordinisane i kooridinisane faze)
- Postojanje više stabilnih stanja (na primer dva jednako dobra izvora hrane)

Tipovi interakcije među socijalnim insektima:

1. Direktna interakcija (razmena tečnosti i hrane, vizuelni kontakt, hemijski kontakt (feromoni))
2. Indirektna interakcija - stigmergija (eng. Stigmergy) - individualno ponašanje menja okruženje, koje posle izaziva promenu ponašanja drugih individua. Na primer, kod mrava, stigmergija eliminiše potrebu za direktnom međusobnom komunikacijom. Efekat je da mravi sprovode koordinisane aktivnosti bez obraćanja jedan drugom kao što to rade ljudi.

Stigmergija je forma indirektne komunikacije koja se vrši modifikacijama sredine. Pojedinci prihvataju signal, koji pokreće određeni odgovor ili akciju. Akcija može da pojača ili modifikuje signal kako bi uticala na akcije drugih pojedinaca. Postoje dva tipa stigmergije: komunikacija putem promene fizičkih karakteristika okruženja (npr. pri pravljenju gnezda, čišćenja gnezda, pojedinac vrši akciju na osnovu toga kakva je tekuća struktura gnezda), ili komunikacija putem mehanizma signala (npr. pri traganju za hranom mravi prate tragove feromona drugih mrava). U algoritmima koji modeluju ponašanje mrava ovakva saradnja pojedinaca se postiže eksploatacijom mehanizama komunikacije koji su proučavani u stvarnim kolonijama mrava. Veštačka stigmergija je definisana kao indirektna komunikacija posredovana numeričkim modifikacijama stanja životne sredine koja su dostupna samo agentima u komunikaciji. Treba naći matematički model koji precizno opisuje stigmergijske karakteristike odgovarajućih pojedinaca. Osnova takvog modela je definicija stigmergijskih varijabli koje enkapsuliraju informacije koje koriste veštački mravi da komuniciraju indirektno. Pri traganju za hranom ulogu stigmergijske varijable ima veštački feromon. Pri traganju za hranom tokom vremena kraći putevi će imati veću koncentraciju feromona, jer se mravi brže vraćaju kad koriste takve puteve. Feromon vremenom isparava, koncentracija feromona na dužim stazama brže se smanjuje nego na kraćim stazama. Veštački feromon oponaša karakteristike pravog feromona i ukazuje na popularnost rešenja za problem optimizacije koji se razmatra. Veštački feromon zapravo kodira dugoročno pamćenje o čitavom procesu pretrage.

Rešavanje problema inteligencijom rojeva. Neke od popularnijih primena su:

- Optimizacija ruta
- Klasterovanje i sortiranje
- Podela posla
- Kooperativni transport
- Izgradnja složenih struktura (gnezda)

Primer: Optimizacija ruta mravima (TSP)
Tražimo najkraći put između dva čvora u grafu. Na početku nema koncentracije feromona na lukovima. Inicijalizujemo koncentraciju feromona na svakoj vezi na nasumičnu malu vrednost. Postavlja se određeni broj agenata (m) na izvorni čvor. Za svaku iteraciju svaki mrav inkrementalno konstruiše put (rešenje) do odredišnog čvora. U svakom čvoru svaki mrav odlučuje sledeći korak u putu, na osnovu tragova feromona ostavljenih od strane drugih mrava. Pokazalo se da mravi brzo konvergiraju ka rešenju i da se malo vremena provodi u istraživanju alternativnih putanja. Kako bi se mravi naterali da više istražuju, i kako bi se sprečila preuranjena konvergencija koncentracija feromona na vezama može da isparava. Konstanta p određuje brzinu isparavanja feromona, čime čini da mravi 'zaborave' prethodne odluke. Za velike vrednosti p feromon brzo isparava, za male sporo. Sa većim isparavanjem pretraga postaje sve više nasumična, a to znači bolje istraživanje. d_{ij} = udaljenost između gradova i i j
τ_{ij} = količina feromona na luku (i, j)
m agenata (mrava) - svaki gradi nezavisnu putanju
U svakom koraku, verovatnoća odlaska od grada i do grada j je srazmerna $(\tau_{ij})^a (d_{ij})^{-b}$
Feromon isparava po formuli: $\tau(1-p)\tau, p \in [0, 1]$

Primer: Rutiranje u komunikacionim mrežama
Agenti započinju svoj put od polaznog čvora ka ciljnom. Svaki agent ažurira svoju tabelu rutiranja i komunicira sa ostalima. Ideja: 'Ako ideš ka ciljnom čvoru u kojem sam ja već bio ranije, daću ti savet kuda da ideš'. Uticaj agenta tj. validnost saveta se smanjuje sa starenjem. Agenti se veštački usporavaju na zagušenim čvorovima (granama) – simulacija realnosti.

Primer: Pomeranje mrava u pravcu hrane (klasterovanje)
Izolovana hrana ima veću šansu da bude pokupljena od strane agenta koji ne nosi tovar. Verovatnoća uzimanja tovara: $P_p = [k_1/(k_1 + f)]^2$, f - gustina hrane u datoj okolini. Agent koji nosi tovar ima veću šansu da ispusti tovar ukoliko u blizini postoje i drugi tovari: $p_d = [f/(k_2 + f)]^2$.

Podela posla. Efikasnost rada postiže se podelom posla među radnicima kolonije, gde je svaki radnik zadužen za podskup potrebnih zadataka. Podela zadataka i koordinacija se odvijaju bez centralne kontrole, posebno za velike kolonije. Podela zadataka je dinamička, zasnovana na unutrašnjim i spoljašnjim faktorima. Iako su jedinke specijalizovane za određene zadatke, promena dužnosti se može desiti kada okolnosti iz životne sredine to zahtevaju – klimatske promene, dostupnost hrane. Jedinke iste starosti uglavnom vrše iste dužnosti. Takođe, dužnosti mogu da se dodeljuju i u skladu sa veličinom jedinke – npr. manji mravi vode računa o mravinjaku, veći tragaju za hranom. *Messor barbarous* mravi koji žive u jugoistočnoj Španiji, donose hranu od izvora ka gnezdu u brigadama od do šestoro radnika. Najpre najmanji mravi uzimaju hranu sa izvora i nose je duž puta dok ne sretnu veće radnike. Veći radnici preuzimaju hranu i nose je dalje, dok se manji vraćaju nazad do izvora. Sličnu organizaciju ima *Tacko Bell.*

Kooperativni transport. Kada pojedinačni mrav ne može da nosi veliki komad hrane, nekoliko mrava se aktivira. Tokom početnog perioda, mravi menjaju poziciju bez očiglednog napretka. Nakon nekog vremena, uspevaju da pomere plen i onda nastavljaju da rade sličnu aktivnost koja daje rezultate. Kolektivna robotika: Naučnici su uspeli da reprodukuju kolektivnu koordinaciju sa grupom veoma jednostavnih robota. Roboti su zajednički gurali kutiju. Možda nije najefikacniji način, ali je potencijalno fleksibilan i moguć da se prilagođava najrazličitijim okolnostima (pod uslovom da se pravila definišu adekvatno).

Izgradnja složenih struktura. Agenti se pomeraju nasumično unutar 3D mreže. Agent postavlja ćeliju (ciglicu) svaki put kad pronađe stimulativnu konfiguraciju. Postoji tabela pravila za stimulativne konfiguracije. Prostor mogućnosti stimulativnih izvedenih konfiguracija je ogroman. Neka pravila mogu biti opasna. Zabeležen je slučaj da su mravi ratnici primenom pravila međusobnog praćenja formirali 'krug smrti'. Krug je bio obima 400 metara i svakom mravu je trebalo oko 2 i po sata da ga obiđe. Tokom nekoliko dana veliki broj mrava je uginuo jer nisu mogli da izađu iz kruga. Nekoliko mrava je ipak uspelo da se izbavi iz stimulativnog feromonskog traga i da razbije krug.

8 Optimizacija rojevima čestica

8.1 Optimizacija rojevima čestica - opšti koncepti i osnovni algoritam

Optimizacija rojevima čestica (eng. Particle swarm optimization – PSO) ima korene u socijalnoj psihologiji. Rojevi čestica su na neki način slični celularnim automatima (eng. Celularautomata - CA):

- Svaka pojedinačna ćelija ažurira svoje stanje paralelno sa ostalima
- Svaka nova vrednost neke ćelije zavisi od starih vrednosti i od vrednosti svojih suseda
- Sve ćelije se ažuriraju primenom istog pravila (Rucker, 1999)

Čestice unutar roja se mogu poistovetiti sa ćelijama unutar CA, samo se njihova stanja menjaju u mnogo dimenzija istovremeno.
Prema objašnjenju autora metode James Kennedy i Russell Eberhart:

"Čestice unutar roja imitiraju socijalno ponašanje ljudi (ili insekata). Čestice (jedinke) interaguju među soboom dok uče iz sopstvenog iskustva, što postepeno pomera populaciju u pravcu boljih regiona rešenja problema".

Zašto ime"čestica", a ne "tačke"? Obojica autora su bili mišljenja da brzine i ubrzanja više priliče česticama nego tačkama.
PSO primene: problemi u realnom, diskretnom ili mešovitom prostoru pretrage; problemi sa višestrukim lokalnim optimumima, ograničenjima; problemi višeciljne, dinamičke optimizacije,...
Originalni PSO:

$$\vec{v_i} \leftarrow \vec{v_i} + \vec{\varphi_1} \otimes (\vec{p_i} - \vec{x_i}) + \vec{\varphi_2} \otimes (\vec{p_g} - \vec{x_i})$$
$$\vec{x_i} \leftarrow \vec{x_i} + \vec{v_i}$$

$\vec{x_i}$ je trenutna pozicija i-te čestice u roju, $\vec{v_i}$ označava brzinu i-te čestice, $\vec{p_i}$ je najbolje pronađena pozicija koju je pronašla i-ta čestica do sada tj. *lični najbolji*, $\vec{p_g}$ je najbolja pozicija u susedstvu tj. *globalni najbolji*, simbol $\otimes$ označava vektorski proizvod, $\vec{\varphi_1} = c_1\vec{r_1}$, $\vec{\varphi_2} = c_2\vec{r_2}$, $\vec{r_1}$ i $\vec{r_2}$ su dva vektora slučajnih brojeva iz ravnomerne raspodele [0, 1], c_1 i c_2 su koeficijenti ubrzanja.
Brzina $\vec{v_i}$ (udeo promena) i-te čestice je definisan sa 3 komponente:

- *momenat* - prethodno stanje brzine se prenosi na novo
- *kognitivna komponenta* - tendencija vraćanja u lično najbolje
- *socijalna komponenta* - tendencija ka kretanju ka najboljem globalnom

Različite topologije se mogu koristiti za usmeravanje toka informacija između čestica, npr. topologija prstena, zvezde, fon Nojmanova topologija.

```
random_initial_population();
repeat
    for i=1 to population_size do
        if f(Xi) < f(Pi) then Pi = Xi;
        Pg = min(Pneighbours);
        for d=1 to dimensions do
            velocity_update();
            position_update();
        end
    end
until termination_criterion_met()
```

Inercijalna težina. Vrednosti $\vec{p_i}$ i $\vec{p_g}$ se mogu prebaciti u jednu vrednost $\vec{p}$ bez gubitka informacija:

$$\vec{v_i} \leftarrow \vec{v_i} + \vec{\varphi} \otimes (\vec{p} - \vec{x_i})$$

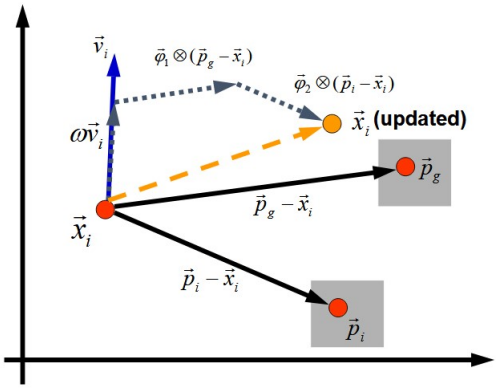
$$\vec{x_i} \leftarrow \vec{x_i} + \vec{v_i}$$

gde je $\vec{\varphi} = \vec{\varphi_1} + \vec{\varphi_2}$ i $\vec{p} = \frac{\vec{\varphi_1} \otimes \vec{p_i} + \vec{\varphi_2} \otimes \vec{p_g}}{\vec{\varphi_1} + \vec{\varphi_2}}$, $\vec{p}$ predstavlja ponderisani prosek $\vec{p_i}$ i $\vec{p_g}$. Primetiti da je operator deljenja po elementima. Za kontrolisanje uticaja prethodne brzine na novu brzinu, može se koristiti dodatni parametar koji se zove *inercija* (inercijalna težina) W:

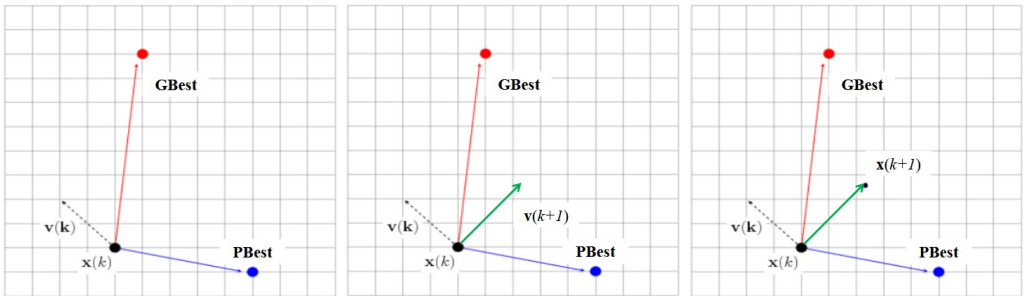
$$\vec{v_i} \leftarrow W\vec{v_i} + \vec{\varphi_1} \otimes (\vec{p_i} - \vec{x_i}) + \vec{\varphi_2} \otimes (\vec{p_g} - \vec{x_i})$$

Inercija omogućava kontrolu eksploracije i eksploatacije. Za $W \geq 1$ brzina raste tokom vremena, pa roj divergira. Za $0 < w < 1$ čestice usporavaju pa konvergencija zavisi od vrednosti c_1 i c_2. Za $W < 0$ brzina se smanjuje tokom vremena, na kraju dostiže 0 i time se zaustavlja algoritam.
Empirijski rezultati ukazuju da konstantna inercijalna težina od $W = 0.7298$ i $c_1 = c_2 = 1.49618$ daju dobre rezultate. Eberhart i Shi takođe savetuju upotrebu inercije koja se smanjuje tokom vremena, obično u rasponu između 0.9 i 0.4. Ovo utiče na smanjenje prostora pretrage tokom vremena i postepeno prebacivanje iz režima jače eksploracije ka režimu jače eksploatacije.

8.2 Geometrijska interpretacija optimizacije rojevima čestica i primeri



PSO numerički primer:



Putanja čestice. Koliko su bitne interakcije između čestica unutar PSO? Da bi odgovorili na ovo pitanje, posmatrajmo jednostavni PSO, i slučaj kada je roj sveden na samo 2 čestice.

Ovako pojednostavljeni PSO pretpostavlja da: nema stohastičke komponente, postoji jedna dimenzija, unapred su određene početne pozicije i brzine.

$$v \leftarrow Wv + c_1(p_i - x) + c_2(p_g - x)$$

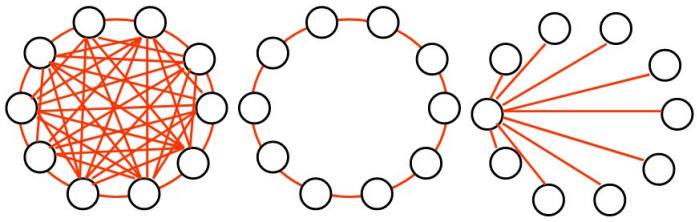
$$x \leftarrow x + v$$

U narednim primerima, protpostavljamo da je $W = 0.7$ i $c_1 = c_2 = 0.7$. Čak i u slučaju samo jedne čestice, znamo pozicije x i p_i. Neka je funkcija koju posmatramo $f(x) = x^2$, definisana na [-20, 20]. Imamo dva slučaja:

1. Prve dve pozicije su na istoj strani minimuma ($x = -20, v = 3.2$)
2. Prve dve pozicije okružuju minimum (x = -2, v = 6.4)

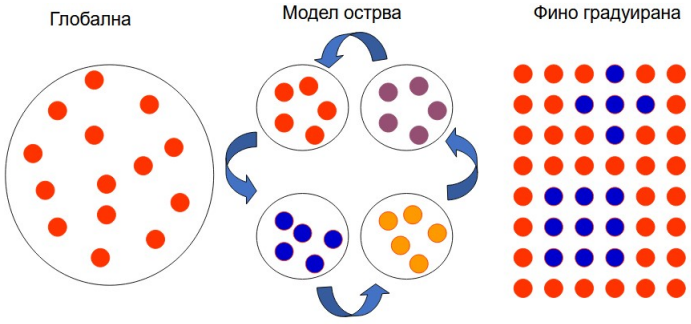
Putanja jedne čestice. Slučaj 1: prve dve pozicije su na istoj strani minimuma - pošto je lični najboji uvek jednak x, čestica nikada neće moći da dostigne minimum (prerana konvergencija). Slučaj 2: prve dve pozicije okružuju minimum - čestica oscilira oko minimuma pošto lični najbolji nije uvek x, što rezultuje boljim ponašanjem.
Putanja dve čestice. Graf uticaja - imamo dva istraživača i dve memorije. Svaki istraživač prima informacije od obe memorije, ali informiše samo jendu memoriju. Početne pozicije su iste kao u slučajevima 1 i 2 ranije. Međutim, čestice sada rade zajendo. Opštiji slučaj je kada je svaka čestica pod uticajem tuđe memorije samo povremeno. Konvergencija ka globalnom optimumu je tada verovatnija, mada ceo proces može biti sporiji.

8.3 Varijante gbest i lbest algoritma i topologije uticaja



Dva najčešća upotrebljavana modela:

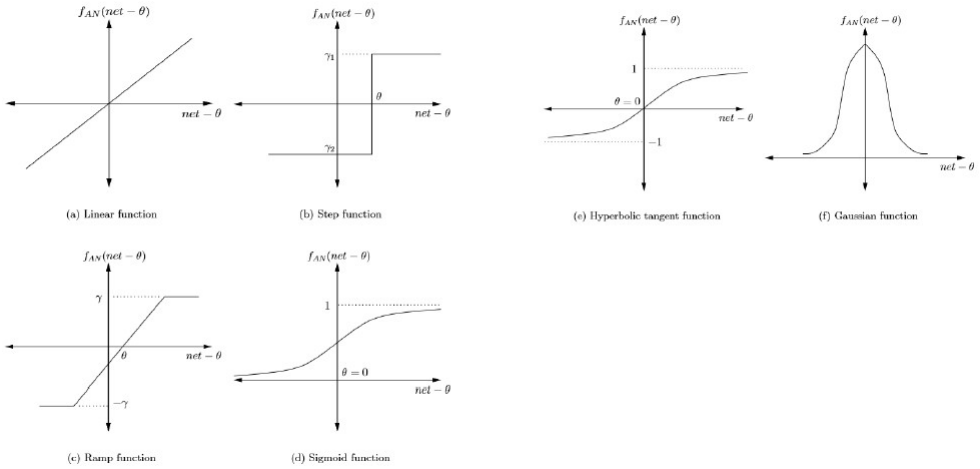
- *gbest* - svaka čestica je pod uticajem najbolje jedinke iz čitavog roja
- *lbest* - svaka čestica je pod uticajem najboljih jedinki iz neke svoje lokalne okoline



Koju koristiti? Praviti kompromis između eksploatacije i eksploracije. *gbest* model najbrže širi informaciju širom populacije. *lbest* model koji koristi topologiju prstena najsporije. Za složene višemodalne funkcije, brza propagacija nije poželjna. Međutim, sporije širenje informacija usporava konvergenciju. Fon Nojmanova topologija se najbolje ponaša.

9 Veštačke neuronske mreže

9.1 Funkcija aktivacije

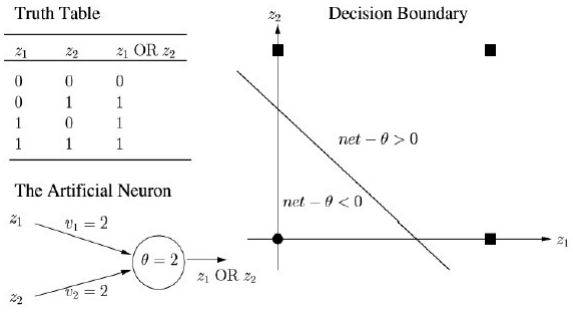


(a) Linear function (b) Step function (e) Hyperbolic tangent function (f) Gaussian function

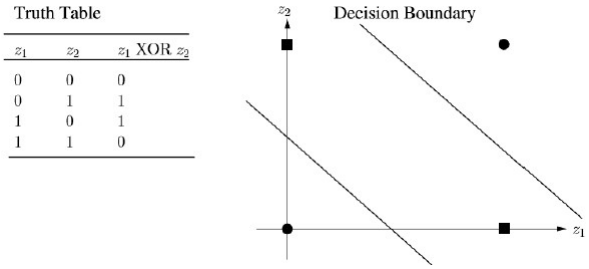
(c) Ramp function (d) Sigmoid function

9.2 Linearna i nelinearna razdvojivost

Linearna razdvojivost - omogućava linearnu razdvojivost bez greške. Postavljanje hiperravni koja razdvaja ulazne podatke na one sa izlazom ispod i iznad nekog praga. Slika prikazuje hiperravan koja odgovara funkciji logičke disjunkcije:



Nelinearna razdvojivost - realizacija eksluzivne disjunkcije zahteva postojanje središnjeg sloja sa dva neurona. Ovo je primer problema koji nije linearno razdvojiv:



9.3 Učenje veštačkog neurona

Učenje gradijentnim spustom - veštački neuron aproksimira funkciju opisanu ulazno izlaznim signalima podešavanjem težina v i parametra θ. Skalarni parametar θ se može pridružiti vektoru težina v radi elegantnije notacije. Aproksimacija se svodi na minimizaciju ukupne greške:

$$\epsilon = \sum_{p=1}^{P_r} (t_p - o_p)^2$$

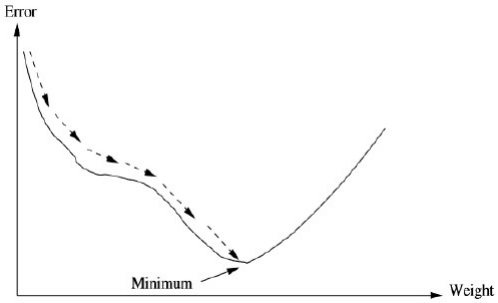
gde t_p i o_p predstavljaju redom ciljnu i aproksimiranu vrednost izlaznog signala, a P_r broj podataka sprovedenih na ulaz. Pravilo gradijentnog spusta omogućava iterativno ažuriranje težina i definisano je kao:

$$v_i(t) = v_i(t-1) + \Delta v_i(t)$$

$$\Delta v_i(t) = \eta(-\frac{\partial \epsilon}{\partial v_i}), \frac{\partial \epsilon}{\partial v_i} = -2(t_p - o_p)z_{i,p}$$

$$v_i(t) = v_i(t-1) + 2\eta(t_p - o_p)z_{i,p}$$

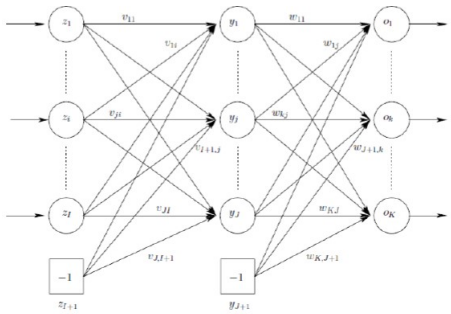
gde je η parametar brzine učenja.



9.4 Tipovi i organizacija veštačkih neuronskih mreža, slike sa objašnjenjima

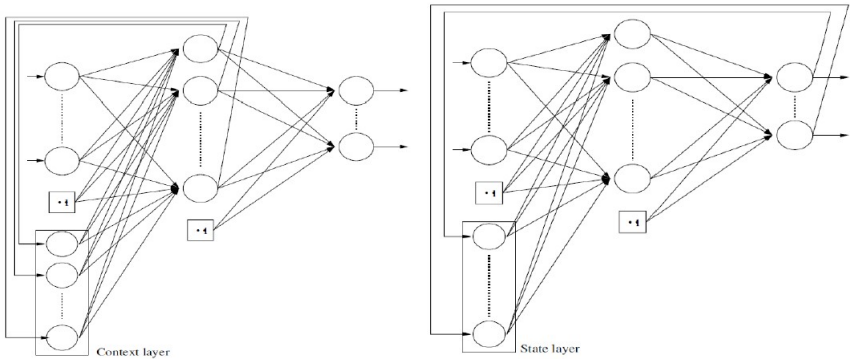
Učenje neuronskih mreža. Pojedinačni veštački neuron omogućava učenje samo linearno razdvojivih funkcija. Grupisanje neurona u mreže to omogućava. Učenje ovakvih mreža je i značajno kompleksnije i računarski zahtevno. Imamo nadgledano i nenadgledano učenje. Nadgledano učenje zahteva skup podataka za trening. Svaki podatak (vektor promenljivih) ima svoju pridruženi ciljnu promenljivu.

Mreže sa propagacijom unapred (FFNN - Feedforward neural network) - najmanje tri sloja: ulazni, srednji i izlazni. Izlaz se računa pomoću jednog prolaska kroz mrežu.

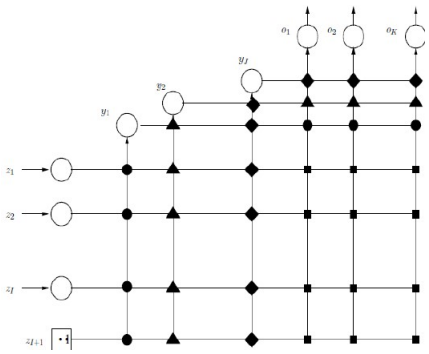


Rekurentne neuronske mreže:

- *Elman SRNN* - Simple recurrent neural network - kopija skrivenog sloja se vraća na ulaz (kontekstni sloj). Cilj je upotreba prethodnog stanja mreže. Omogućava npr. učenje temporalnih zavisnosti.
- *Jordan SRNN* - kopija izlaznog sloja se sprovodi na ulaz (tzv. sloj stanja)



Kaskadne neuronske mreže (CNN - Cascade NN) - svi ulazi spojeni sa svim skrivenim i svim izlaznim elementima. Elementi srednjeg sloja spojeni sa svim izlazima i svim narednim elementima srednjeg sloja.



9.5 Pravila nadgledanog i nenadgledanog učenja

Pravila nadgledanog učenja. Neka je dat konačan skup uređenih parova ulaznih vrednosti i pridruženih ciljnih vrednosti: $D = \{dp = (z_p, t_p)|p = 1, ..., P\}$, gde su $z_{i,p}, t_{k,p} \in R$ za $i = 1, ..., I$ i $k = 1, ..., K$. I je broj ulaznih signala, K je broj izlaznih signala, P je broj trening podataka. Tada se može predstaviti sledeća zavisnost: $t_p = \mu(z_p) + \xi p$, gde je $\mu(*)$ nepoznata ciljna funkcija, a ξp šum.
Cilj učenja je aproksimirati datu funkciju $\mu(*)$ na osnovu podataka iz D. Polazni skup D se obično deli na tri disjunktna podskupa:

- D_T - trening skup za aproksimaciju
- D_V - skup za validaciju (memorizacija)
- D_G - skup za testiranje (procena kvaliteta uopštavanja)

Tokom faze učenja minimizuje se empirijska greška podešavanjem W:

$$\epsilon_T(D_T; W) = \frac{1}{P_T}\sum_{p=1}^{P_T}(F_{NN}(z_p, W) - t_p)^2$$

Postoje razne tehnike za optimizaciju ovog tipa - metode lokalne optimizacije (gradijentni spust) i metode globalne optimizacije (metaheuristike).
Izazovi: preprilagođavanje i potprilagođavanje.
Gradijentni spust za učenje NN - sastoji iz dve faze:

1. Propagacija signala unapred, jednostavno računanje signala za FFNN
2. Propagacija greške unazad: signal greške se šalje nazad ka ulaznom sloju pri čemu se vrši izmena težinskih koeficijenata

Neka je suma kvadratna greška (eng. Sum squared error - SSE) uzeta za funkciju cilja minimizacije: $\frac{1}{2}\sum_{k=1}^{K}(t_k - o_k)^2$, i neka se koristi sigmoidna funkcija aktivacije i na izlaznom i na središnjem sloju: $o_k = f_{o_k}(net_{o_k}) = \frac{1}{1+e^{-net_{o_k}}}$.
Stohastički gradijentni spust za učenje NN - težine se ažuriraju na sledeći način:

$$w_{kj}(t) += \Delta w_{kj}(t) + \alpha\Delta w_{kj}(t-1)$$

$$v_{ji}(t) += \Delta v_{ji}(t) + \alpha\Delta v_{ji}(t-1)$$

gde je α tzv. momenat koji definiše značaj prethodne promene.

$$\Delta w_k j = \eta(-\frac{\partial E}{\partial w_{kj}}) = -\eta\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial w_{kj}} = -\eta\delta_{o_k}y_j$$

gde je:

$$\delta_{o_k} = \frac{\partial E}{\partial net_{o_k}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_{o_k}} = -(t_k - o_k)(1-o_k)o_k = -(t_k - o_k)f'_{o_k}$$

$$\frac{\partial o_k}{\partial net_{o_k}} = \frac{\partial f_{o_k}}{\partial net_{o_k}} = (1-o_k)o_k = f'_{o_k}$$

Slično i za ažuriranje težina između ulaznog i srednjeg sloja:

$$\Delta v_{ji} = \eta(-\frac{\partial E}{\partial v_{ji}}) = -\eta\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_{ji}} = -\eta\delta_{y_j}z_i$$

gde je:

$$\delta_{y_j} = \frac{\partial E}{\partial net_{y_j}} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial net_{y_j}} = \sum_{k=1}^{K}\delta_{o_k}w_{kj}f'_{y_j}$$

$$\frac{\partial y_j}{\partial net_{y_j}} = \frac{\partial f_{y_j}}{\partial net_{y_j}} = (1-y_j)y_j = f'_{y_j}$$

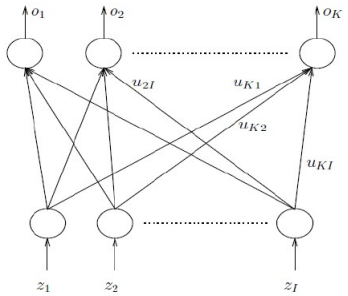
Algoritam:

```
Inicijalizuj tezine i broj epoha t = 0;
while nije zadovoljen uslov za zavrsetak do
    E = 0;
    for svaki trening podatak p do
        Propagiraj podatak unapred i racunaj y i o;
        Racunaj signale gresaka delta_o i delta_y;
        Azuriraj tezine w i v (propagacija gre aka unazad);
        E += Ep
    end
    t = t + 1
end
```

Nenadgledano učenje. Kod nadgledanog učenja, koristi se pristup u kojem se mreži daje ulaz i očekivani izlaz (poput „učitelja“). Na osnovu ovoga, greška biva „kažnjavana“ ažuriranjem težina u slučaju da postoji greška. U suprotnom se ne radi ništa. Kod nenadgledanog učenja ne postoji očekivani izlaz. Algoritam učenja mora samostalno da utvrdi postojanje pravilnosti u ulaznim podacima. Veštačke neuronske mreže omogućavaju pravljenje asocijacija između šablona (eng. Pattern association). Ovakve mreže se još zovu i asocijativna memorija ili asocijativne neuronske mreže.

9.6 Asocijativna neuronska mreža i hebovo učenje

Asocijativne neuronske mreže - su obično dvoslojne. Cilj je da omoguće stvaranje asocijacije bez upotrebe „učitelja“. Razvoj ovakvih mreža zasnovan je na studijama vizuelnog i zvučnog korteksa kod mozga sisara. Topološka organizacija neurona omogućava asocijaciju. Dodatna poželjna karakteristika je zadržavanje starih informacija i nakon pristizanja novih (nadgledanim učenjem ovo obično ne može da se postigne).
Primer asocijativne mreže: Funkcija koju uči ovakva mreža je preslikavanje ulaznog šablona u izlazni, $f_{NN} : R^I \to R^K$



Hebovo učenje - nazvano po neuropsihologu Hebb-u. Težine se ažuriraju na osnovu korelacije između aktivacionih vrednosti neurona. Zasnovano na hipotezi: „potencijal neurona da ispali signal je zavistan od od potencijala okolnih neurona“. Težina između dva korelisana neurona se pojačava:

$$u_{ki}(t) = u_{ki}(t-1) + \Delta u_{ki}(t), \Delta u_{ki}(t) = \eta o_{k,p} z_{i,p}$$

Izmena težine je veća za one ulazno izlazne parove kod kojih ulazna vrednost ima jači efekat na izlaznu vrednost. Problem je što ponovno ubacivanje ulaznih šablona dovodi do eksponencijalnog rasta težina. Ovo se rešava postavljanjem limita na vrednost težina. Primer limita je nelinearni faktor zaboravljanja:

$$\Delta u_{ki}(t) = \eta o_{k,p} z_{i,p} - \gamma o_{k,p} u_{ki}(t-1)$$

gde je γ pozitivna konstanta koja kontroliše umanjenje.

9.7 Kvantizacija vektora 1

LVQ-1 klasterovanje (Learning Vector Quantizer-1) - nenadgledana metoda učenja za klasterovanje. Cilj je skup od n podataka grupisati u m grupa: tako da su elementi iz iste grupe slični međusobno. Za meru sličnosti različitosti se obično koristi Euklidsko rastojanje. Izlazne vrednosti (oznake klastera) se „takmiče“ za ulazne podatke. Algoritam:

```
Inicijalizuj tezine mreze, brzinu ucenja i precnik susedstva;
while nije zadovoljen uslov za zavrsetak do
    for svaki ulazni podatak p do
        Izracunaj Euklidsko rastojanje, d, izmedju ulaznog vektora z i svakog
    vektora tezinu u;
        Pronadji izlaznu vrednost o za koju je rastojanje d najmanje;
        Azuriraj sve tezine u susedstvu k formulom (*);
    end
    Azuriraj bazu ucenja;
    Smanji precnik susedstva;
end
```

9.8 Samoorganizujuće mape

Samoorganizujuće mape (Self organizing feature maps - SOM) - razvio ih je Kohonen u nameri da modelira karakteristike ljudskog celebralnog korteksa. Metoda vrši projekciju I-dimenzionog ulaznog prostora u izlazni diskretni prostor (neki vid kompresije). Izlazni prostor je često dvodimenziona mreža vrednosti. Ideja je zadržavanje topologije strukture ulaznog prostora. Ako su dva podatka blizu u ulaznom prostoru, biće blizu i u izlaznom. Slične moždane aktivnosti aktiviraju bliske neurone.
SOM - stohastičko pravilo učenja - zasnovano na kompetitivnoj strategiji učenja. Vrlo slično LVQ-1 klasterovanju. Ulazni podaci su povezani sa odgovarajućim neuronima u mapi. Mapa je obično kvadratnog oblika. Broj neurona je manji od broja trening podataka. U idealnom slučaju broj neurona je jednak broju nezavisnih trening primeraka. Vektor težina za svaki neuron na poziciji (k, j) je inicijalno nasumično podešen: $w_{kj} = (w_{kj1}, w_{kj2}, ..., w_{kjI})$. Svaki ulazni podatak je povezan sa svakim neuronom iz mape. Primetiti da je dimenzija vektora težina ista kao i dimenzija ulaznog podatka. Za svaki pročitani podatak sa ulaznog sloja, pronalazi se neuron iz mape koji ima najsličniji težinski vektor. Sličnost može npr. biti Euklidska. Nad tim „pobedničkim“ neuronom vrši se korekcija težina u skladu sa ulaznim podatkom. Takođe se vrši korekcija težina susednih neurona proporiconalno njihovoj udaljenosti od „pobednika“. Kako odmiče trening, smanjuje se i opseg susednih neurona i na samom kraju se smatra da neuroni više nemaju suseda.
Primene SOM: analiza slika, prepoznavanje zvuka, procesiranje signala, telekomunikacije, analiza vremenskih serija. Pogodnosti: omogućava laku vizuelizaciju i intepretaciju, oblasti koje klasifikuju (kategorišu) su vidljive na mapi.