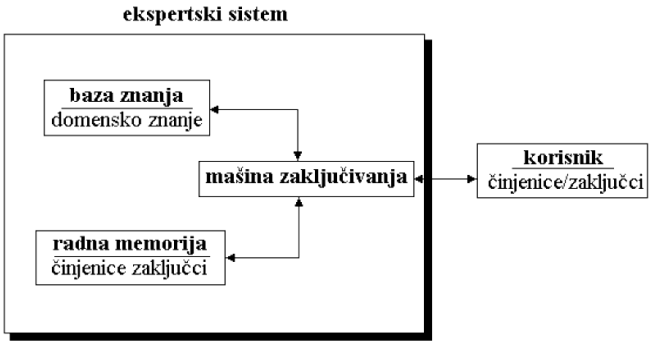
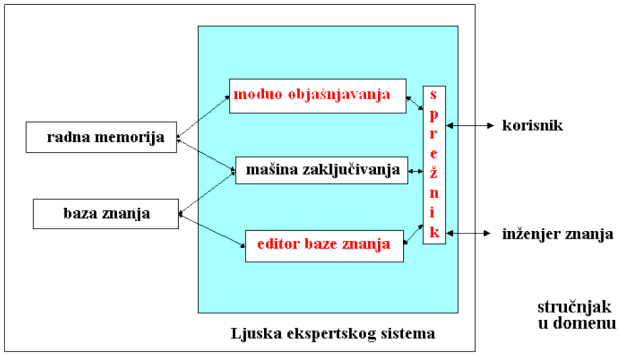
Uvod

Veštačka inteligencija - oblast računarskih nauka koja se bavi oponašanjem čovekovih mentalnih funkcija u računaru
Inteligentni sistemi se bave analizom i projektovanjem autonomnih sistema koji mogu imati senzore i aktuatore i mogu biti ugrađeni u druge sisteme, a poseduju sledeće karakteristike:
- računarski oponašaju čovekova inteligentna svojstva
- pokazuju racionalno ponašanje radi izvršenja zadataka
- sarađuju sa drugim sistemima ili čovekom

Ekspertski sistemi

Ekspertski sistem - računarski program koji emulira rešavanje problema na način na koji to čini ekspert
Pravilo - struktura znanja kojom se povezuje poznata informacija sa drugom informacijom koja se može zaključiti i tako učiniti poznatom
Za predstavljanje činjenica mogu se koristiti okviri - forme za predstavljanje znanja o nekom objektu, objektu se dodaje ime atributa i njegova vrednost





Osobine
- odvojeno znanje od upravljanja
- kodiranjem obuhvaćeno stručno znanje sadrži kako domensko znanje tako i veštine rešavanja problema u domenu
- fokusiranje stručnosti
- rasuđivanje korišćenjem simbola
- heurističko rasuđivanje
- ograničeni na rešive probleme
- netačno rasuđivanje

Delovi
Baza znanja (knowledge base) - sadrži znanje o domenu
Radna memorija (working memory) - sadrži činjenice o razmatranom problemu otkrivene tokom sesije korišćenja sistema
Mašina zaključivanja (inference engine) - upoređuje činjenice sadržane u radnoj memoriji sa domenskim znanjem sadržanim u bazi znanja na specifičan način radi dobijanja zaključaka o razmatranom problemu
- mašina zaključivanja je zadužena za upravljanje radom sistema i obradu pravila
- razdvajanje baze znanja od mašine zaključivanja, uz iskazni karakter znanja omogućava konstruisanje i održavanje baze
- ne postavlja se pitanje šta program treba da radi, već šta treba da zna

Neizvesne cinjenice
- korišćenje faktora izvesnosti - numeričke vrednosti stepena poverenja
- korišćenje fuzzy činjenica

Aktiviranje pravila
- Ekspertski sistem koristi pravila iz baze znanja i činjenice iz radne memorije radi rešavanja problema
- kada je AKO deo pravila jednak informaciji nađenoj u radnoj memoriji, izvršava se ONDA deo pravila i to se naziva aktiviranjem pravila
- iskaz iz ONDA dela pravila se dodaje u radnu memoriju kao zaključena činjenica, što može aktivirati neko drugo pravilo (ovaj postupak se naziva ulančavanje)

Ulančavanje unapred (forward chaining), data- driven search, recognize-resolve-act

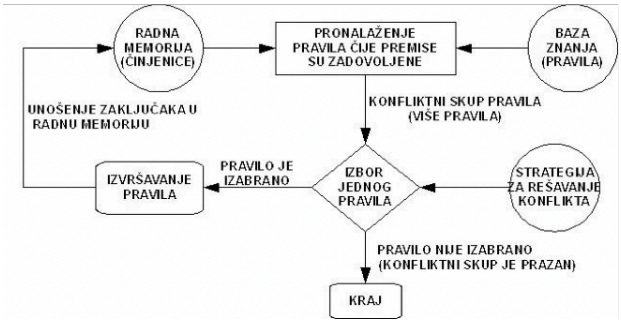
- polazi od poznatih činjenica
- izvodi nove činjenice koristeći pravila čije premise se poklapaju sa poznatim činjenicama
- nastavlja ovaj proces do dostizanja ciljnog stanja ili do ustanovljavanja da nema više pravila čije premise se poklapaju sa poznatim ili izvedenim činjenicama
- zaključuje sve moguće informacije iz raspoloživih informacija (bez obzira na važnost)

Prednosti

- prilagođeno problemima čije rešavanje počinje prikupljanjem informacija, pa zaključivanjem
- može generisati veliku količinu novih informacija
- pogodno za probleme planiranja, nadzora, upravljanja, interpretiranja

Nedostaci

- ne pravi razliku između informacija po važnosti
- može postavljati zbunjujuća pitanja



Ulančavanje unazad (backward chaining), goal- driven search

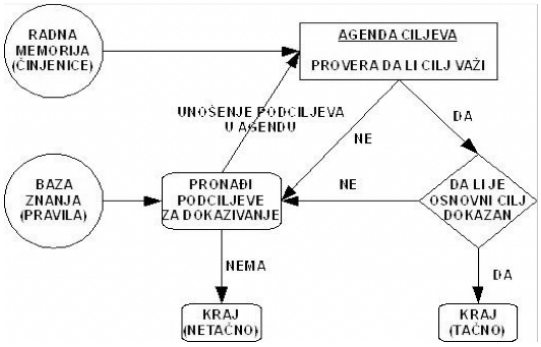
- strategija zaključivanja koja pokušava da dokaže hipotezu prikupljajući informacije koje tu hipotezu podržavaju
- počinje rad od cilja kojeg treba da dokaže i proverava da li cilj postoji u radnoj memoriji
- ako cilja nema u radnoj memoriji, sistem pretražuje pravila, tražeći ono, ili ona, koja sadrže cilj u ONDA delu pravila
  * taj tip pravila se naziva pravilo cilja
- dalje proverava da li su premise pravila cilja prisutne u radnoj memoriji
  * one premise koje nisu prisutne u radnoj memoriji postaju novi ciljevi (podciljevi), za koje se dokazuje da mogu biti podržani drugim pravilima
  * agenda ciljeva - hijerarhijska struktura ciljeva koje je potrebno dokazati da bi se dokazao osnovni cilj
- proces se nastavlja rekurzivno, dok sistem ne nađe premisu koja nije podržana ni jednim pravilom (premisa pravila koja nije zaključak ni u jednom drugom pravilu) - ova premisa se naziva primitiv
  * informaciju o primitivu sistem traži od korisnika, a koristi je za dokazivanje podciljeva i cilja

Prednosti

- prilagođeno problemima čije rešavanje počinje formiranjem hipoteze
- fokusirano na dati cilj
- pretražuje samo deo znanja u vezi postavljenog cilja

Nedostaci

- nastaviće rasuđivanje iako je cilj zamenjen nekim drugim



Kako stručnjak rešava problem?
- ako se prvo prikupljaju podaci, a onda zaključuje: ulančavanje unapred
– ako se postavlja hipoteza o rešenju, pa se traže dokazi za hipotezu: ulančavanje unazad

Prostor pretraživanja:
- više podataka, manje zaključaka: ulančavanje unapred
- manje podataka, više zaključaka: ulančavanje unazad

Mehanizam za objašnjavanje
ZAŠTO - objašnjenje o tome zašto postavlja određeno pitanje
KAKO - objašnjenje o tome kako je stigao do rešenja
STRATEGIJA - koju je strategiju izabrao da bi stigao do rešenja (meta-pravila, heuristike)

Tragpravila (ruletrace)
- lista naziva izvršenih pravila u redosledu izvršavanja
- lista trenutnijh činjenica koje su dovele do izvršenja svakog pravila (trenutno stanje radne memorije)
- primer: Pravilo 1 → Pravilo 4 → Pravilo 2 → KRAJ

Učaureni tekst (templates, cannedtext)
- unapred utvrđene rečenice koje mogu da imaju i dinamičke delove, npr. vrednosti promenljivih
- primer: Ako je napon na akumulatoru <X> što je manje od optimalnih 12V, akumulator je prazan

Objašnjenje strategije
- objašnjavaju se koraci (znanje,metapravila,meta heuristike) koji su usmerili proces zaključivanja
- meta-pravila, zajedno sa grupama pravila omogućavaju optimizaciju procesa zaključivanja fokusiranjem na pravila koja najviše obećavaju
- meta-pravila ne vode uvek tačnom rešenju
- primer: Ako auto neće ni da vergla, usmeriti rešavanje problema na probleme sa električnim sistemom auta

Tjuringov test
- testiranje se sastoji u pisanoj online konverzaciji ispitivača sa mašinom u trajanju od 5 minuta
- ako 30% vremena mašina uspeva da zavara ispitivača, kažemo da je prošla test
- fokus je na racionalnom ponašanju

Inteligentni agenti

Agent - bilo šta što se može posmatrati kao da opaža svoje okruženje koristeći senzore i što deluje na okruženje preko aktuatora
Agent - sistem koji se ponaša racionalno tako da ostvaruje najbolji ishod i podrazumeva:
- rad pod autonomnim upravljanjem
- opazanje okuzenja
- uportnost u dugacnom vremenskom periodu
- adaptiranje na promene
- sposobnost usvajanje drugacijih ciljena

Agent i Tjuringov test
- sposobnost da predstavi znanje i da rasudjuje pomocu njega
- sposobnost da generise razumljive recenice u prirodnom jeziku
- konstantno obucavanje
- vizuelna percepcija

Opažaj - opažajni ulazi agenta (dobijeni preko senzora) u bilo kom datom trenutku
Opažajna sekvenca agenta - potpuna istorija svega što je agent ikada opazio
Agentov izbor akcije u bilo kom datom trenutku može zavisiti od čitave opažajne sekvence uočene do tog trenutka, ali ne i od bilo čega što on nije opazio
Ponašanje agenta je opisano funkcijom agenta
Funkcija agenta - apstraktan matematički opis - specificira akciju kao odziv na opazajnu sekvencu
Program agenta - konkretna realizacija funkcije agenta, koja se izvršava u nekom fizičkom sistemu

Specifikacija okruženja - sadrži meru performanse, spoljno okruženje, aktuatore i senzore

Mera performanse agenta

Mera performanse - kriterijum uspeha za ponasanje agenta
- racionalni agent bira onu akciju koja maksimizuje očekivanu vrednost mere performanse za do tog trenutka datu sekvencu opažaja
- Sveznajuci (omniscience) agent zna stvarni ishod svojih akcija i može delovati shodno tome, a to u realnosti nije moguce
- Racionalnost maksimizuje očekivane performanse, dok perfekcija maksimizira stvarne performanse
- Agent je autonoman ukoliko je njegovo ponašanje odredjeno sopstvenim iskustvom (sa sposobnošću učenja i adaptacije)

Šta je racionalno u bilo kom datom trenutku, zavisi od četiri stvari:
- mere performanse kojom se definiše kriterijum uspeha
- agentovog prethodnog znanja o okruženju
- akcija koje agent može izvršavati
- agentove opažajne sekvence do posmatranog trenutka

Definicija racionalnog agenta
- Za svaku moguću opažajnu sekvencu, racionalni agent treba da izabere delovanje za koje se očekuje da maksimizuje njegovu meru performanse, za činjenice date opažajnom sekvencom i za bilo koje ugrađeno znanje koje agent ima

POAS opis (neophodno za projektovanje agenta)
- Mera **P**erformanse: bezbednost, odredište, zarada
- **O**kruženje: ulice, pešaci, vreme
- **A**ktuatori: upravljanje, gas, kočnice
- **S**enzori: merači ubrzanja, senzori motora, GPS

Opseg okruzenja zadataka

Potpuno opservabilno / delimično opservabilno
* ako senzori agentu daju pristup potpunom stanju okruženja u svakoj tački u vremenu, onda kažemo da je okruženje zadatka potpuno opservabilno

Determinističko / stohastičko
* ako je sledeće stanje okruženja potpuno određeno trenutnim stanjem i akcijom koju vrši agent, onda kažemo da je okruženje determinističko

Epizodičko / sekvencijalno
* u epizodičkom okruženju zadatka, iskustvo agenta deli se u atomske epizode gde svaku epizodu čini agent koji opaža i potom vrši jednu akciju i sledeca epizoda ne zavisi od akcija koje su preduzete u prethodnim epizodama

Statičko / dinamičko
* ako se okruženje može menjati dok agent promišlja, onda kažemo da je okruženje dinamičko
* ako se okruženje ne menja dok agent promišlja, ali se menja vrednost mere performanse agenta okruženje je poludinamičko

Diskretno / kontinualno
* ako promena okruženja predstavlja neprekidnu funkciju, onda kažemo da je okruženje kontinualno

Jednoagentno / višeagentno
* ako je u istom trenutku na svetu prisutno vise objekata koji se mogu smatrati agentima, onda je to viseagentno okruzenje
* ako je ponašanje objekta A najbolje opisano maksimiziranjem mere performanse, čija vrednost zavisi od ponašanja agenta B, onda se objekat A moze smatrati agentom
* moze biti potpuno i delimicno kompetitivno i kooperativno

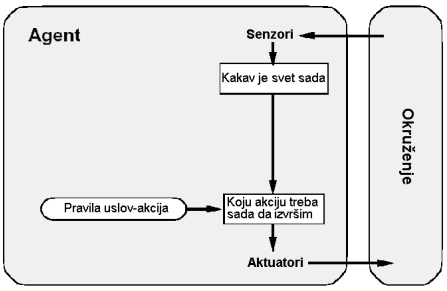
Poznato / nepoznato
* ako agent zna kako funkcionise okruzenje u kom se nalazi, onda je to poznato okruzenje
* ne odnosi se samo na okruzenje, vec i na stanje znanja agenta

Arhitektura agenta (agent = arhitektura + program)
- agent moze kao ulaz uzeti samo trenutni opazaj, a moze uzeti i citavu istoriju opazaja

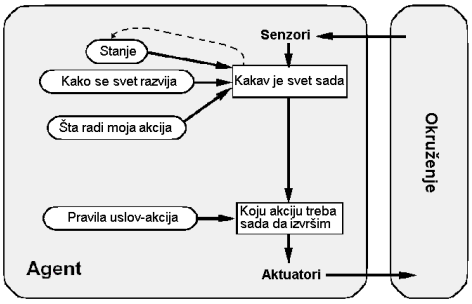
Tabelom upravljan agent
- agent belezi opazajnu sekvencu i potom je koristi kao indeks za tabelu akcija da bi odlucio sta da radi
- glavni nedostatak ove arhitekture je velicina tabele

Tipovi agenata

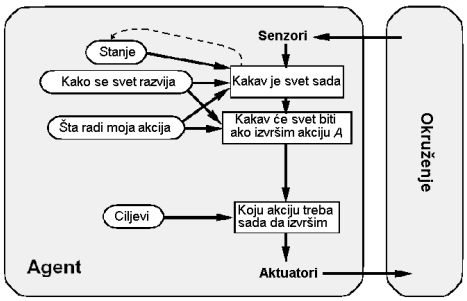
- jednostavni refleksni agenti
    * biraju akcije na osnovu trenutnih opazaja ignorisuci preostalu istoriju opazaja
    * ovi agenti imaju smisla samo ako je okruzenje potpuno observabilno



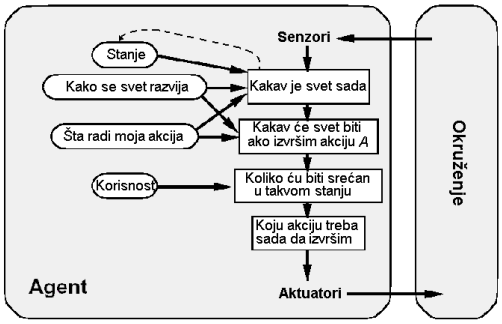
- refleksni agenti sa stanjem (agenti zasnovani na modelu)
    * delimična observabilnost se može uspešno tretirati tako što će agent pratiti promene onog dela okruženja koje mu je nedostupno
    * kodovanje 2 vrste znanja
        - Kako se svet razvija nezavisno od agenta?
        - Kako delovanje agenta utice na svet?



- agenti zasnovani na cilju
    * znanje o trenutnom stanju okruzenja nije uvek dovoljno pa je agentu potrebna ciljna informacija kojom se opisuju situacije koje su pozeljne
    * lako je ako se cilj postize jednom akcijom, a u suprotnom pretrazivanje i planiranje pomazu



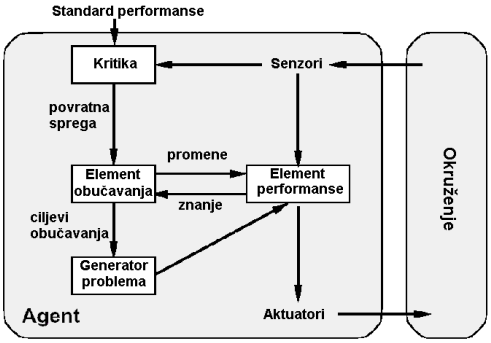
- agenti zasnovani na korisnosti
    * pokusavaju da maksimizuju svoju "srecu"
    * funkcija korisnosti agenta je internalizacija mere performansi
    * ako su unutrašnja funkcija korisnosti i spoljna mera performansi u saglasnosti, onda će agent koji bira akcije maksimizirajući svoju korisnost biti racionalan u skladu sa spoljnom merom performansi
    * u delimično observabilnom i stohastičnom okruženju, racionalni agent zasnovan na korisnosti bira akciju koja maksimizuje očekivanu korist



- agenti sa obucavanjem
    * element obucavanja - odgovoran za generisanje poboljsanja

- element obučavanja koristi povratnu spregu iz kritike u vezi sa delovanjem agenta, i odredjuje kako treba modifikovati element performanse da agent bolje ubuduće radi

* element performanse - odgovoran za delovanje na osnovu opazanja
* obučavanje kod inteligentnih agenata se može posmatrati kao proces modifikacije svake komponente agenta radi dovodjenja komponente u veću saglasnost sa raspoloživom informacijom dobijenom povratnom spregom, čime se poboljšava ukupna performansa agenta



Projektovanje i realizacija
- Proceduralni pristup - koriscenje programskog jezika
- Deklarativni pristup - definise univerzalni jezik komunikacije agenta

Predstavljanje okruzenja (osa porasta izrazajnosti)
- Atomsko
    * svako stanje sveta je nedeljivo
- Faktorisano
    * svako stanje ima atribute
    * atributi ne moraju imati vrednost
- Struktuirano
    * svako stanje ima atribute koji su medjusobno povezani

Neinformisano pretrazivanje

Agenti rešavanja problema (jedna vrsta agenata zasnovanih na cilju) koriste atomska predstavljanja, gde se stanja sveta se posmatraju kao celine bez unutrašnje strukture i odlučuju šta da rade tako što pronalaze sekvence akcija koje vode poželjnim stanjima
Agenti zasnovani na cilju koriste naprednija predstavljanja (faktorisano, strukturirano) i obično se nazivaju agentima planiranja
Neinformisani algoritmi (uninformed search, blind search) - nemaju nikakve informacije o problemu osim same definicije; jedino mogu da generisu sledbenike i da razlikuju ciljno stanje od stanja koje to nije
Informisani algoritmi - imaju nekakva znanja o tome gde da traže rešenja
Formulacija cilja je proces u kojem se odlucuje u kojim stanjima je cilj postignut
Formulacija problema je proces u kojem se za dati cilj odlucuje koje akcije i stanja treba razmatrati
Prostor stanja problema formira graf u koje cvorovi predstavljaju stanja, a grane izmedju njih su akcije
Resenje je fiksirana sekvenca akcija koja vodi do cilju
Kvallitet resenja se meri funkcijom troska putanje, a optimalno resenje je ono sa najnizim troskom putanje od svih resenja

Okruzenje
Delimicno opservabilno - agent uvek zna trenutno stanje, ali ne vidi cilj
Diskretno - u svakom datom stanju postoji konacno mnogo akcija izmedju kojih se moze izvrsiti izbor
Poznato - agent zna u koje stanje ga vodi svaka akcija
Deterministicko - svaka akcija ima tacno jedan ishod

Pretrazivanje
Problem se moze formalno opisati pomocu 5 komponenti:
1. **Pocetno stanje** - konkretno stanje od kojeg agent pocinje sa resavanjem problema
2. **Opis mogucih akcija** - skup uredjenih parova (akcija, sledbenik) gde je akcija jedna od dozvoljenih akcija u trenutnom stanju, a sledbenik je stanje u koje se moze doci iz trenutnog stanja ako se primeni ta akcija
3. **Model prelaza (transition model)** - opis toga sta svaka akcija radi; rezultat akcije; isto sto i "sledbenik" iz (akcija, sledbenik)
4. **Testiranje cilja** - utvrdjuje da li je dato stanje ciljno stanje; eksplicitno definisan skup ciljnih stanja ili apstraktno stanje npr. "sah-mat"
5. **Funkcija troska putanje** - svakoj putanji dodeljuje numericki iznos troska koji predstavlja meru performanse; trosak(trenutno stanje, akcija, naredno stanje)

Apstrakcija

Apstrakcija je postupak uklanjanja detalja iz predstave problema (stanja i akcije se apstrahuju)
Apstrakcija je **validna** ako se svako apstraktno resenje moze prosiriti u resenje u detaljnijem svetu
Apstrakcija je **korisna** ako je izvodjenje svake akcije u resenju lakse od prvobitnog problema
Ilustrativni problem (toy problem) namenjen je kao ilustracija ili vezba za razlicite metode resavanja problema
Problem iz realnog sveta (real world problem) je problem cije resavanje ima znacaja za ljude

Raspored skladistenja umetnutih cvorova

1. **FIFO (first in first out, red)**
2. **LIFO (last in first out, stek)**
3. **Priority queue**

Performanse

Potpunost - da li algoritam garantovano pronalazi resenje ako ono postoji
Optimalnost- da li strategija pronalazi optimalno resenje (najnizi trosak)
Vremenska slozenost - koliko vremena je potrebno
Prostorna slozenost - koliko memorije je potrebno
Konzistentnost - da li algoritam svaki put za isti ulaz daje isti izlaz

Mere slozenosti

Vreme se meri brojem cvorova generisanih tokom pretrazivanja
Prostor se meri maksimalnim brojem cvorova koje se cuvaju u memoriji
Trosak pretrazivanja - kolicina vremena utrosena za pretrazivanje
Ukupan trosak - kombinuje trosak pretrazivanja i trosak putanje pronadjenog resenja
b (faktor grananja) - maksimalan broj sledbenika nekog cvora
d (dubina najpliceg ciljnog cvora) - broj koraka duz putanje od korena
m (dubina najdubljeg cvora) - maksimalna duzina putanje u prostoru
C* (optimalni trosak) - trosak optimalnog resenja
e (trosak akcije) - minimalni trosak jedinicke akcije
L (granicni nivo) - nivo posle koga se pretrazivanje zaustavlja

Strategije neinformisanog pretrazivanja

1. Pretrazivanje u sirinu (BFS)
2. Pretrazivanje sa uniformnim troskom
3. Pretrazivanje u dubinu (DFS) + vracanje
4. Pretrazivanje sa ogranicenom dubinom
5. Iterativno pretrazivanje u dubinu sa povecavanjem dubine
6. Dvosmerno pretrazivanje

Pretrazivanje u sirinu (BFS)

Svi cvorovi na datoj dubini stabla se razvijaju pre razvijanja bilo kog cvora na sledecem nivou
Koristi se **FIFO** red cekanja (queue)
Testiranje cilja se vrsi pri generisanju cvora, a ne pri biranju cvora za razvijanje
Jeste potpuno - ako je najplici ciljni cvor na nekoj konacnoj dubini, pretrazivanje u sirinu ce ga na kraju pronaci poso prosiri sve plice cvorove (pod uslovom da je faktor grananja konacan)
Nije optimalno - pretrazivanje u sirinu je optimalno ako je trosak putanje neopadajuca funkcija dubine cvora (troskovi svih koraka su jednaki)
Vreme - $O(b^d)$
Prostor - $O(b^d)$

Pretrazivanje sa uniformnim troskom

Umesto da prosiruje najplici cvor (BFS), algoritam prosiruje cvor sa najnizim troskom putanje
Ne obraca paznju na broj koraka, vec samo na ukupan trosak
Koristi se **red cekanja sa prioritetom** (priority queue)
Testiranje cilja se vrsi pri biranju cvora za razvijanje, a ne pri generisanju cvora zato sto prvi ciljni cvor koji se generise moze biti na suboptimalnoj putanji (plus je dodat test za slucaj da je nadjena bolja putanja ka cvoru koji je trenutno u rubu)
Nije potpuno - potpunost moze da se garantuje samo ako je trosak svakog koraka veci ili jednak nekoj maloj pozitivnoj konstanti
Jeste optimalno - troskovi su nenegativni pa putanje nikada ne postaju krace dodavanjem cvorova
Vreme - $O(b^{1+[C^*/e]})$
Prostor - $O(b^{1+[C^*/e]})$

Pretrazivanje u dubinu (DFS)
Uvek prosiruje najdublji cvor u rubu
Koristi se **LIFO** red cekanja (stack)
Testiranje cilja se vrsi pri biranju cvora za razvijanje
Jeste i nije potpuno - za graf je potpuno ako ima konacan broj stanja, za stablo je nije potpuno zbog beskonacnih petlji
Nije optimalno - ako postoje dva cilja na razlicitim dubinama, naci ce se onaj dublji
Vreme - $O(b^m)$
Prostor - O(bm)

Pretrazivanje sa vracanjem
Svaki delimicno prosireni cvor pamti koji sledbenik treba sledeci da se generise
Koristi se **LIFO** red cekanja (stack)
Testiranje cilja se vrsi pri biranju cvora za razvijanje
Jeste potpuno - ne pamti se referenca na roditelja
Nije optimalno - ako postoje dva cilja na razlicitim dubinama, naci ce se onaj dublji
Vreme - $O(b^m)$
Prostor - O(m)

Pretrazivanje sa ogranicenom dubinom
Postojanjem granice resava se problem neogranicenog stabla
Koristi se **LIFO** red cekanja (stack)
Testiranje cilja se vrsi pri biranju cvora za razvijanje
Nije potpuno - ako je nivo cilja veci od granicnog nivoa, pretrazivanje nece uspeti
Nije optimalno - ako izaberemo l<d, pretrazivanje nije optimalno
Vreme - $O(b^L)$
Prostor - O(bL)

Pretrazivanje u dubinu sa iterativnim produbljivanjem (PIP)
Pronalazi najbolju granicu dubine tako sto je postepeno povecava. Prekida kada granica postane d. Kombinuje BFS i DFS. Cvorovi u najdubnljem nivou se generisu samo jednom, a njih ima najvise pa algoritam ne ispada toliko skup.
Koristi se **LIFO** red cekanja (stack)
Testiranje cilja se vrsi pri biranju cvora za razvijanje
Jeste potpuno - za graf je potpuno ako ima konacan broj stanja, za stablo je nije potpuno zbog beskonacnih petlji
Jeste optimalno - troskovi su nenegativni pa putanje nikada ne postaju krace dodavanjem cvorova
Vreme - $O(b^d)$
Prostor - O(bd)

Iterativno pretrazivanje sa produzavanjem (IPP)
Nasladjuje pretrazivanje sa uniformnim troskom i slicno kao PIP samo sto povecava granicu troska umesto granicu dubine.

Dvosmerno pretrazivanje
Koristi se BFS i pocetnog i ciljnog stanja. Moguce je samo kada se zna gde je ciljni cvor. Zahteva vise prostora zbog hes tabele.
Koristi se **FIFO** red cekanja (queue) i **hes tabela** za proveru da li je cvor razvijen pretragom iz suprotnog smera.
Testiranje cilja se vrsi pri generisanju cvora, a ne pri biranju cvora za razvijanje
Jeste potpuno - za graf je potpuno ako ima konacan broj stanja, za stablo je nije potpuno zbog beskonacnih petlji
Jeste optimalno - troskovi su nenegativni pa putanje nikada ne postaju krace dodavanjem cvorova
Vreme - $O(b^{d/2})$
Prostor - $O(b^{d/2})$

kriterijum	u širinu	uniform. troška	u dubinu	ogran. dubine	iter.pov. dubine	dvosmer.
Potpun?	Da [a]	Da [a, b]	Ne	Ne	Da [a]	Da [a, d]
Vreme	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Prostor	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal.?	Da [c]	Da	Ne	Ne	Da [c]	Da [c, d]

Informisano pretrazivanje

Funkcija vrednovanja f(N) - za prosirivanje se bira cvor sa najnizom procenom troska
Heuristicka funkcija h(N) - procenjeni trosak najjednostavije putanje do ciljnog stanja, zavisi samo od stanja u cvoru N
Trosak putanje g(N) - trosak putanje od pocetnog cvora do cvora N
Metodi informisanog pretraživanja mogu imati pristup heurističkoj funkciji h(n), koja daje procenu troška od čvora n
Generički algoritam pretraživanje prvo najbolji bira čvor za proširivanje u skladu sa funkcijom vrednovanja
Performanse heurističkih algoritama pretraživanja zavise od kvaliteta heurističke funkcije

* Ponekad je moguće napraviti dobru heuristiku **ublažavanjem** definicije problema, prethodnim izračunavanjem troškova rešenja za **podprobleme** u jednoj bazi podataka oblika ili **obučavanjem** iz iskustva tako sto se reši mnogo problema

Greedy best-first search
Pokusava prvo da prosiri cvor koji je najblizi cilju
Koristi samo heuristicku funkciju za prcenu troska => f(N) = h(N)
Nije potpuno - moze da krene niz beskonacnu petlju
Nije optimalno
Vreme - $O(b^m)$
Prostor - $O(b^m)$

A* search
Identicno pretrazivanju sa uniformnim troskom samo sto se dodaje heuristika
Cvorovi se procenjuju pomocu kombinacije troska stizanja do cvora N i procenjenog troska od N do cilja => f(N) = g(N) + h(N)
Koristi **postupak odsecanja** kako ne bi razmatrao sve mogucnosti
Jeste potpuno
Jeste optimalno
Optimalno efikasan - nijedan drugi optimalni algoritam neće garantovano proširiti manji broj čvorova od A*
Vreme - $O(b^{\varepsilon d})$
Prostor - $O(b^{\varepsilon d})$
Apsolutna greška se definiše kao $\Delta = h^* - h$, gde je h^* stvarni trošak stizanja iz korena do cilja
Relativna greška se definiše kao $\varepsilon = (h^* - h)/h^*$

Uslov optimalnosti
1. Heuristika mora biti **prihvatljiva** - nikad ne precenjuje trosak stizanja do cilja
   - Prihvatljive heuristike su po prirodi optimisticne prosto smatraju da je trosak resavanja problema manji nego sto stvarno jeste
2. Heuristika mora biti **konzistentna/monotona** - postuje nejednacinu trougla
   - Heuristika je konzistentna ako za svaki cvor N i svakog sledbenika N' od N generisanog akcijom A, procenjeni trosak dostizanja cilja iz N nije veci od troska stizanja u N' plus procenjeni trosak stizanja iz N' do cilja => $h(n) \leq c(n,a,n') + h(n')$
   - Za prihvatljivu heuristiku, nejednakost čini savršeni smisao: ako postoji put iz *n* u cilj *G*n najbliži čvoru *n,* preko *n*' koji je jeftiniji od *h*(*n*), to bi narušilo svojstvo da je *h*(*n*) donja granica troška dostizanja *G*n

Memorijski ograniceno heuristicko pretrazivanje
Za A* algoritam prvo ponestane prostora mnogo pre nego vremena zato sto se cuvaju svi generisani cvorovi
Algoritmi koji stede memoriju:
1. Algoritam iterativnog produbljivanja A* (IDA)
2. Rekurzivno pretrazivanje prvo najbolji (RPPN)
3. Memorijski ograniceno A* (MA*)
4. Pojednostavljeno memorijski ograniceno A* (PMA*)

Algoritam iterativnog produbljivanja A* (IDA)
* Razlika izmedju IDA i PIP: kao granica se koristi f-trosak, a ne dubina; kod svake iteracije granica odsecanja je najnizi f-trosak svih cvorova koji su u prethodnoj iteraciji bili van granice
* Pati od istih teskoca sa realno procenjenim troskovima kao i iterativna verzija pretrazivanja sa uniformnim troskom
* Mana algoritma je sto zaboravlja veci deo onoga sto je uradio

Rekurzivno pretrazivanje prvo najbolji (RPPN)
* Pokusava da imitira rad standardnog pretrazivanja prvo najbolji, ali koristi samo linearni prostor
* Struktura algoritma je slicna rekurzivnom pretrazivanju po dubini, ali umesto da beskonacno nastavlja po trenutnoj putanji, algoritam koristi promenljivu f-granica za stalno cuvanje f vrednosti najbolje alternativne putanje dostupne od bilo kog pretka trenutnog cvora - ako trenutni cvor predje ovu granicu rekurija se vraca na alternativnu putanju
* RPPN pamti f-vrednost najboljeg lista u zaboravljenom podstablu - kada prosiruje najbolji cvor pamti f-vrednost drugog najboljeg cvora

* Mana je to sto ponovo generise iste cvorove (**brbljanje** / **trashing**)
Jeste potpuno
Jeste optimalano ako je heuristika prihvatljiva
Vreme - zavisi od preciznosti heuristike i od toga koliko cesto se najbolja putanja menja tokom prosirivanja
Prostor - O(bd)
* Mana algoritma je sto zaboravlja veci deo onoga sto je uradio

Pojednostavljeno memorijski ograniceno A* (PMA*)
* Prosiruje cvorove kao A* sve dok se memorija ne popuni, a u tom trenutnku izbacuje neki cvor i dodaje trenutni
* Uvek izbacuje cvor sa najvecom f-vrednoscu
* Kao RPPN, pravi rezervnu kopiju vrednosti zaboravljenog cvora kod njegovog roditelja i na taj nacin moze da se vrati i da prosiri zaboravljene putanje ako se sve ostale putanje pokazu gorim
* Da bi se izbeglo biranje istog cvora i za brisanje i za prosirivanje, algoritam prosiruje najnoviji najbolji list, a brise najstariji najgori list
Jeste potpun
Jeste optimalan
* Mana je to sto ponovo generise iste cvorove (**brbljanje** / **trashing**)
* Sto je manja memorija, to je potrebno vise vremena za dostizanje cilja

Obucavanje sa ciljem boljeg pretrazivanja
Metanivoski prostor stanja - svaki cvor metanivoskog prostora stanja je jedan ceo graf koji predstavlja stanje programa koji vrsi pretrazivanje
Cilj obucavanja je minimizacija ukupnih troskova resavanja problema, sa trazenjem balansa izmedju troska izracunavanja i troska putanje

Heuristike
Heuristika h2 dominira nad heuristikom h1 ako h2 daje efikasnije resenje (kracu putanju) od h1
Heuristicke funkcije se porede na osnovu efektivnog faktora grananja
Efektivni faktor grananja b* - faktor granjanja koje bi stablo dubine d moralo imati da bi sadrzalo resenje
Dobro projektovanoj heuristici bi vrednost b* bila blizu 1
Graf prostora ublazenog problema je supergraf pocetnog prostora jer se uklanjanjem restrikcija kreiraju dodatne grane u grafu
Trosak optimalnog resenja za ublazeni problem je **prihvatljiva** heuristika za prvobitni problem (optimisticna heuristika)
Posto je izvedena heuristika tacan trosak za ublazeni problem, ona mora da zadovoljava nejednacinu trougla pa je **konsistentna**
Prihvatljive heuristike se mogu izvesti iz troska resenja **podproblema** (npr. od 8 objekata, samo 4 treba da budu na svom mestu)
Baza podataka uzora - cuvanje tacne cene za sve potprobleme
Razdvojena baza podataka uzora - cuvanje cena disjunktnih poteza daje donju granicu za resavanje celog problema
Heuristika se moze nauciti na osnovu iskustva koje podrazumeva resavanje mnogo problema (neural networks, decision trees, reinforcement learning)
Metodi ucenja najbolje funkcionisu kada dobiju svojstva stanja koja su relevantna za njegovo procenjivanje, a ne samo sirov opis stanja

Lokalno pretrazivanje

* Informisano pretraživanje razmatra jednu kategoriju problema: opservabilna, deterministička, poznata okruženja u kojima je rešenje jedna sekvenca akcija
* Lokalno pretrazivanje je pogodno za probleme u kojima je bitno samo stanje resenja, a ne trosak puta do njega
Onlajn pretrazivanje - agent se suocava sa prostorom stanja koji je u pocetku nepoznat i koji se mora istrazivati
Prednosti: koristi se malo memorije (konstantna velicina), pronalazi se resenje u velikim ili beskonacnim prostorima
Prikaz prostora stanja ima **lokaciju** (x, trenutno stanje) i **elevaciju** (y, funkcija cilja)
Ako elevacija odgovara trosku, tada je cilj pronaci najdublju dolinu - **globalni minimum**
Ako elevacija odgovara funkciji cilja, tada je cilj pronaci najvisi vrh - **globalni maksimum**
Potpuni algoritam uvek pronalazi cilj ako on postoji
Lokalni maksimumi - vrh koji je visi od svih susednim stanja, ali je nizi od globalnom maksimuma
Grebeni - sekvenca lokalnih maksimuma po kojima se gramzivi algoritmi tesko krecu
Plato - ravno podruzje u prikazu prostora stanja; treba ograniciti broj uzastopnih koraka kako se ne bi uslo u beskonacnu petlju

Pretrazivanje usponon (hill-climbing search)
Petlja koja se neprekidno krece u smeru bolje vrednosti i zavrsava se kada dostigne vrh gde nema suseda sa boljim vrednostima
Ne gleda dalje od neposrednih suseda trenutnog stanja
Cesto ostaje u lokalnom ekstremumu i ne nadje globalni
Nije potpuno
Stohasticki uspon - na slucajan nacin bira neki od poteza uzbrdo; verovatnoca izbora varira sa strminom poteza uzbrdo

Uspon prvog izbora - uvodi stohasticki uspon tako sto generise sledbenike na slucajan nacin sve dok generisani sledbenik ne bude bolji od trenutnog stanja

Uspon sa slucajnim ponovnim kretanjem
Izvodi se niz pretrazivanja usponom od slucajno generisanih pocetnih stanja i zaustavlja se kada se pronadje cilj
Ako ima mali broj lokalnim maksimuma i platoa, algoritam ce brzo pronaci resenje
Skoro jeste potpuno

Pretrazivanje simuliranim kaljenjem
1. Algoritam koji samo ide gore nije potpun
2. Algoritam koji nasumicno bira stanje iz skupa sledbenika jeste potpun, ali nije efikasan
Simulirano kaljenje kombinuje ova dva algoritma i u pocetku dozvoljava losija stanja, ali svakim sledecim korakom se verovatnoca za dolazak u takvo stanja smanjuje
$e^{\Delta E/T}$ - verovatnoca da prihvati sledeci potez
ΔE - razlika izmenju troskova trenutnog i narednog stanja
T - vreme proteklo od pocetka pretrage

Lokalno pretrazivanje po snopu
Ovaj algoritam prati K stanja, a ne samo jedno
1. Pocinje sa K slucajno generisanih stanja
2. Na svakom koraku se generisu svi sledbenici svih K stanja
3. Ako je bilo koje od njih cilj, algoritam se zaustavlja
4. Inace, algoritam iz cele liste bira K najboljih sledbenika i ponavlja postupak
Kod pretrazivanja sa slucajnim ponovnim kretanjem svaki proces pretrazivanja se izvrsava nezavisno od ostalih
Kod lokalnog pretrazivanja po snopu, K paralelnih niti pretrazivanja razmenjuje korisne informacije

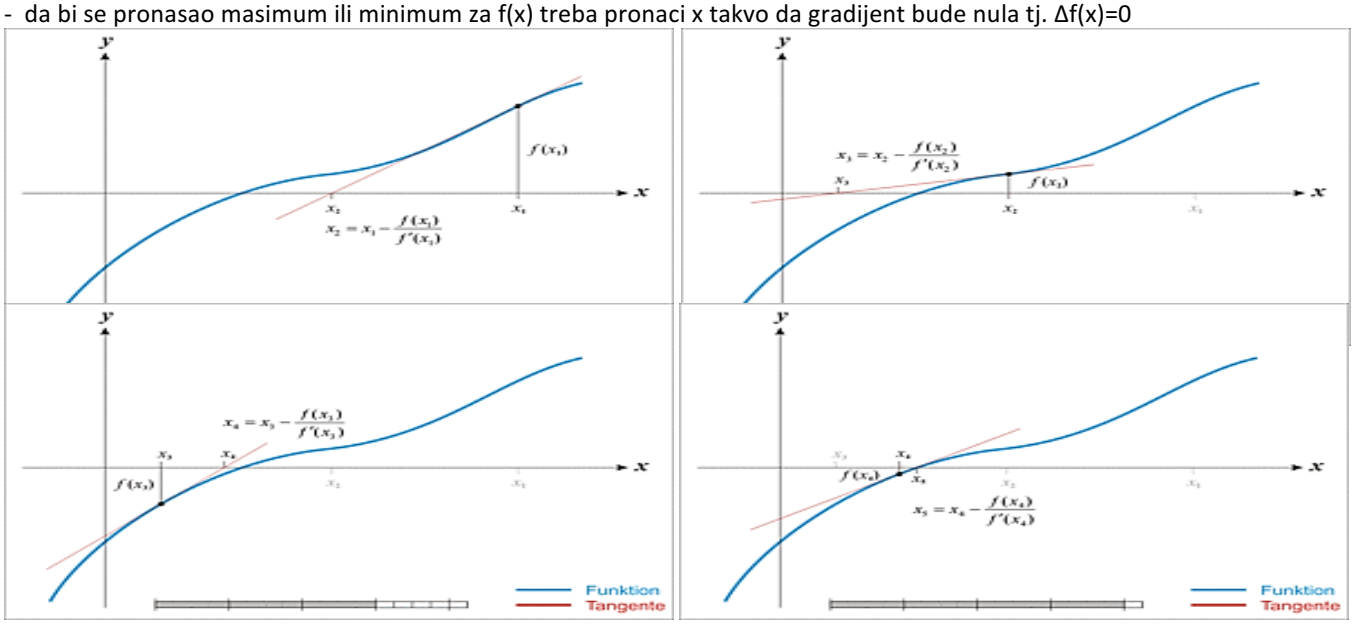
Stohasticko pretrazivanje po snopu
Obicno lokalno pretrazivanje po snopu moze da pati od nedostatka raznolikosti medju K stanja
Kod stohastickog pretrazivanja umesto da se medju kandidatima za sledbenika bira najboljih K, ovde se K sledbenika bira na slucajan nacin tako sto se verovatnoga izbora datog sledbenika povecava sa njegovom vrednoscu - **prirodna selekcija**

Genetski algoritmi
* Varijanta stohastickog pretrazivanja po snopu u kojem se stanja sledbenici genrisu kombinovanjem dva roditeljska, umesto modifikacijom samo jednog stanja (**seksualna reprodukcija**)
* Pocinje skupom od K slucajno generisanih stanja koje nazivamo **populacijom**
* Svako stanje/**jedinka** je niz konacnog niza **gena (genom)**
* Svako stanja/jedinka ima svoju **fitnes vrednost** koja govori o tome koliko je blizu resenju problema
* **Sablon** - podniz genoma u kojem neke pozicije mogu da ostanu nedefinisane
* Jedan sablon moze imati vise **instanci**
* Ako je fitnes vrednost jednog sablona iznad proseka, tada ce se njegov tog sablona u populaciji vremenom povecavati
* Operator **selekcije** se odnosi na metod odabira jedinke koja ce ucestvovati u reprodukciji - bira se na osnovu fitnes vrednosti
* Operator **rekombinacija (crossover)** kreira dva nova potomka nasumicno (bolje je da tacka zavisi od fitnes vrednosti) birajuci tacku kidanja dnk lanca dve jedinke i ukrstanjem tih delova
* Operator **mutacije** nasumicno menja gen u dnk lancu po pravilu sa jako malom verovatnocom (verovatnoca se smanjuje kako vreme odmica da bi se izbegla mutacija dobre populacije)
* Koraci:
1. predstaviti problem kao genome fiksirane duzine, izabrati velicinu populacije i verovatnocu mutacije
2. definisati funkciju fitnesa koja meri performanse svake jedinke
3. nasumicno generisati pocetnu populaciju
4. izracunamo fitnes svake jedinke
5. izaberemo jedinke sa najboljim fitnesom za roditelje zamenimo populaciju sa njihovom decom
6. ponavljamo racunanje fitnesa, selekciju, rekombinaciju i mutaciju sve dok se ne zadovolji zadati kriterijum

Lokalno pretrazivanje u kontinualnim prostorima
* Metodi lokalnog pretrazivanja imaju u neprekidnim prostorima stanja probleme lokalnih maksimuma, lanaca vrhova i platoa isto koliko i u diskretnim prostorima
* Jedno resenje je da pojednostavimo problem tako sto **diskretizujemo** okolinu svakog stanja
* Drugo resenje je da koristimo gradijent
* **Gradijent funkcije cilja** je vektor koji daje velicinu i smer najstrmijeg uspona - tako ce sledece stanje biti na najvecoj visini - parcijalni izvod (fiksiramo odredjene promenljive)
* Mozemo da pronadjemo maksimum tako sto izjednacimo izvod sa nulom (linije je paralelna sa X osom)
* Mozemo da izvrsimo pretrazivanje najstrmijim usponom ako se trenutno stanje x=x+a*Δf(x) gde je a mala konstanta

* Ako je konstanta a previse mala, potrebno je mnogo koraka, a ako je previse velika, pretraga moze da prebaci maksimum
* precedura **linijskog pretrazivanja** duplira vrednost konstante a sve dok f(x) ne pocne opet da opada
* U nekim slucajevima ne postoji funkcija koja moze da se diferencira i tada se moze naci **empirijski gradijent** tako sto se proceni reakcija na mala povecanja i smanjenja svake koordinate
* **Njutnov metod optimizacije:**

- pretraga se izvrsava tako sto se izracuna nova procena resenja x prema jednacini x=x-Δf (x)/ Δf'(x)
- da bi se pronasao masimum ili minimum za f(x) treba pronaci x takvo da gradijent bude nula tj. Δf(x)=0



* Problem optimizacije je ogranicen ako resenja moraju da zadovolje cvrsta ogranicenja za vrednost svake promenljive
* Oflajn agenti prvo izracunavaju potpuno resenje pre nego sto kroce u realan svet, a zatim izvrsavaju resenje

Agent onlajn pretrazivanja

Radi tako sto prepice izracunavanje i akcije pa prvo preduzima akciju, a zatim postra okruzenja i razmatra sledecu akciju
Onlajn pretrazivanje je korisno u **dinamickim** domenima gde postoji kazna za neaktivnost i previse dugo izracunavanje
Onlajn pretrazivanje je korisno u **nedeterministickim** domenima jer dozvoljana agentu da usredsredi svoje racunske napore na nepredvidivost koje se stvarno pojavljuju, umesto na one koje bi se mogle, ali u stvari nece, desiti
Onlajn pretrazivanje je neophodno za **nepoznata** okruzenja u kojima agent ne zna koja stanja postoji i koje posledice imaju njegove akvcije u takvim okruzenjima agent uci u hodu
Agent obicno ima dva cilja: da dodje do ciljnog stanja uz sto manji torsak ili da istrazi celo okruzenje

Odnos konkurentnosti

Trosak putanje koji agent zaista prelazi se poredi sa troskom putanje koju bi agent presao da je unapred poznavao prostor pretrazivanja tj. najkracu putanju (ovaj odnos treba da bude sto moguce manju)
Ako su neke akcije ireverzibilne, onlajn pretrazivanje moze slucajno da dospe u stnanje corsokaka iz kojeg nije dostupnan cilj
Da bise to resilo, treba pretpostaviti da je prostor stanja **bezbedno istraziv** tj. da se iz svakog stanja moze doci do cilja
Cak i u bezbedno istrazivim prostorima ne moze se garantovati ograniceni odnos konkurentnosti ukoliko postoje putanje sa neogranicenim troskom i zbog toga se performanse algoritama onlajn pretrage opisuju u smislu velicine celog prostora stanja, a ne samo prema dubini najpliceg cilja
Onlajn algoritam moze samo da prosiri cvor u kome se fizicki nalazi za razliku od oflajn algoritma koji prosiruje sve cvorove
Da bi se izbegao put preko celog stabla da bi se prosirio sledeci cvor, bolje je da se cvorovi prosiruju u **lokalnom redosledu**

Onlajn agent pretrazivanja u dubinu

Cuva mapu u tabeli gde belezi stanje koje nastaje iz akcije u prethodnom stanju
Kada neka akcija iz trenutnog stanja nije istrazene, agent pokusava da je izvrsi
Teskoca nastupa kad je agent isprobao sve akcije jednog stanja

- u oflajn pretrazivanju u dubinu stanje se izbacuje i reda
- u onlajn pretrazivanju agent treba fizicki da se vrati u prethodno stanje

Agent je **primenljiv** samo na prostore u kojima se svaka akcija moze opozvati nekom drugom akcijom
U najgorem slucaju se tacno dva puta preci svaku vezu u prostoru stanja
Odnos konkurentnosti ovog agenta moze da bude los ako on odluta na dugacak put a postoji cilj odah pored pocetnog stanja

Onlajn lokalno pretrazivanje

Pretrazivanje usponom razvija samo lokalne cvorove u moze da stane nepokretan u lokalnim ekstemumima
Ne mogu se koristiti slucajna ponovna kretanja zato sto agent ne moze sebe da prenese u novo stanje

Moze se razmotriti upotreba **hasumicnog hoda** gde se na slucajan nacin bira jedna od akcija dostupnih iz trenutnog stanja
Prioritet imaju akcije koje nisu isprobane pa tako je pretraga potpuna ali veoma spora

Learning real time A*
Efikasnije je prosiriti uspon memorijom tako sto se za svako poseceno stanje cuva trenutno najbolja procena - peglanje"
Agent ce ici napred-nazad sve dok ne "spljosti" lokalni maksimum azuriranjem heuristike za to stanje i tada preci na suseda
U kontinualnim prostorima ne pronalazi cilj

Obucavanje u onlajn pretrazivanju
Agenti uce ishode svake akcije u svakom stanju
Agenti lokalnog pretrazivanja pribavljaju preciznije procene vrednosti svakog stanja koristeci lokalna pravila (LRTA)
Da bi agent bio pametniji, potreban je formalan prikaz opstih pravila i algoritmi koji izgradjuju opsta pravila na osnovu iskustva

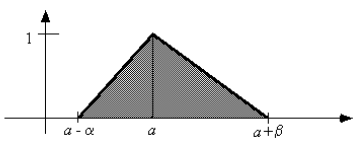
Rasplinuti skupovi

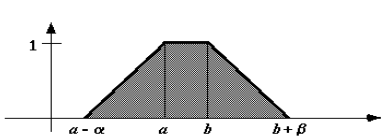
Meko racunarstvo
Oblast VI koja se bavi konstruisanjem sistema za koje se ocekuje da poseduju:
1. strucnost u specificiranom domenu
2. sposobnost obucavanja u promenljivom okruzenju
3. sposobnost objasnjavanja nacina na koji su doneli odredjene odluke

Meko vs tradicionalno racunarstvo
Znanje o posmatranoj pojavi
Tolerancija nepreciznosti
Univerzalna aproksimativnost

Karakteristike mekog racunarstva
1. strucnost - domensko znanje
2. bioloski inspirisani modeli
3. novi modeli optimizacije
4. numericka izracunavanja
5. obucavanje na osnovu podataka uzoraka
6. nalazenje pravila ili regularnosti
7. sistemi sa tolerancijom greski
8. realne primene

* **Rasplinuti podskup** A skupa X moze se definisati kao skup uredjenih parova u svakom od kojih je prvi element iz skupa X, a drugi element is intervala [0, 1], pri cemu tacno jedan uredjeni par odgovara svakom elementu iz skupa X
* **Funkcija pripadanja** je preslikavanje definisano tako sto element iz X slika u vrednost iz intervala [0, 1]
* Neka je X neprazan skup. Rasplinuti skup A u domenu razmatranja X je okarakterisan svojom funkcijom pripadanja koja se interpretira kao stepen pripadanja elementa x rasplinutom skupu A za svako x iz X
$A = \{(x, \mu(x)) \mid x \in X\}$
$\mu: X \rightarrow [0,1]$
* Rasplinuti podskupovi realne prave nazivaju se rasplinute velicine
* Familija svih rasplinutih podskupova u skupu X oznacena je sa F(X)
* **Nosac skupa** A, u oznaci sup(A), je podskup skupa X ciji svi elementi imaju pripadnost vecu od nule
* **Jezgro skupa** A, u oznaci ker(A), je podskup skupa X ciji svi elementi imaju pripadnost jednaku jedinici
* **Srednja tacka** skupa A, u oznaci mid(A), je tacka iz skupa X koja ima pripadnost 0.5
* **Normalan** rasplinuti skup je rasplinuti podskup A skupa X ako postoji element koji ima pripadnost jednaku jedinici
* **Subnormalan** skup je svaki rasplinuti skup koji nije normalan
* **Alfa presek** je skup elemenata koji imaju pripadnost vecu ili jednaku broju alfa; obelezava se sa $[A]^{\alpha}$
* Rasplinuti skup A domena X naziva se **konveksnim**, akko za svako x1 i x2 iz X vazi:
$\mu(\lambda x1 + (1 - \lambda)x2) \geq \min(\mu(x1), \mu(x2)), \lambda \in [0, 1]$
* **Princip razlaganja** - rasplinuti skup A se moze predstaviti pomocu unije alfa preseka
* **Rasplinuti broj** A je rasplinuti skup ciji je domen realna brojevna prava koji ima normalnu, konveksnu i neprekidnu funkciju pipadanja sa ogranicenim nosacem
* $\min[A]^{\alpha}$ - leva strana alfa preseka (funkcija leve strane je monotono rastuca)
* $\max[A]^{\alpha}$ - desna strana alfa preseka (funkcija desne strane je monotono opadajuca)
* Ako je $\alpha \leq \beta$, onda $[A]^{\alpha} \supset [A]^{\beta}$

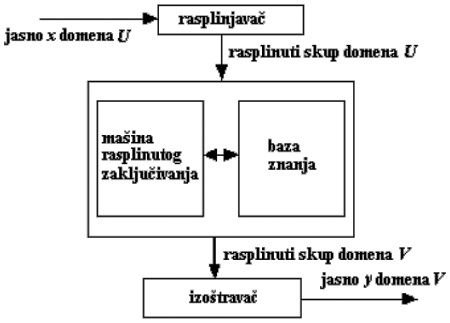




* Rasplinuti skup A ciji je domen razmatrana skup R naziva se **trougaonim brojem** sa centrom a, levom sirinom $\alpha > 0$ i desnom sirinom $\beta > 0$, ako njegova funkcija pripadanja ima sledeci izgled
* Slicno, **trapezasti broj** sa intervalom tolerancije [a, b] ima izlged
* Skup A je **rasplinuti podskup** skupa B ako je vrednost funkcija pripadanja u A manja od iste u B za isti element
* Skupovi A i B su **jednaki** ako je vrednost funkcija pripadanja u A jednaka istoj u B za isti element
* **Prazan rasplinuti podskup** domena X ako funkcija pripadanja ima vrednost 0 za svako x iz X
* **Univerzalni rasplinuti skup** domena X ako funkcija pripadanja ima vrednost 1 za svako x iz X
* Ako nosac skupa ima samo jedan element, onda je on **rasplinuti singlton**
* **Presek** rasplinutih skupova A i B je $\mu(x) = \min(\mu(x), \mu(x)) = \mu(x) \wedge \mu(x), \ \forall x \in X$
* **Unija** rasplinutih skupova A i B je $\mu(x) = \max(\mu(x), \mu(x)) = \mu(x) \vee \mu(x), \ \forall x \in X$
* **Komplement** rasplinutog skupa A je $\mu\neg A(x) = 1 - \mu A(x)$
* Zakoni **iskljucenja treceg** i **nekontradikcije** ne vaze

Regulatori

* Ulazne vrednosti u bazu znanja koju cine rasplinuta pravila treba da budu rasplinuta i zato se uvodi **rasplinavanje**
* Da bi se dobila jasna vrednost izlaza, izlazni rasplinuti skup mora proci kroz **izostravanje**



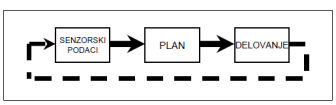
Robotika

- Inteligentni robot je mehanicka tvorevina koja autonomno funkcionise tako sto senzorima primi informaciju o okurzenju, odredi sledecu akciju pomocu aktuatora i naredbu prenese efektorima koji izvrsavaju tu akciju
- Pored fizickih posotje i softverski roboti (softboti)
- Teleoperacija/telemehanika je rad masine na rastojanju od njenog upravljaca (rad na daljinu)
- Tvorevina je ako se o necemu misli kao da postoji (entitet)
- Roboti obavljaju poslove koje ziva bica ne mogu ili ne zele
- Modaliteti: terenska vozila, bespilotne letelice, podvodno vozilo, plovila
- Komponente: pokretljivost (efektori), opazanje (senzori), upravljanje (aktuatori), energija, komunikacija
- Robotika koristi znanje svih 7 oblasti VI, a to su: predstavljanje zanja, razumevanje prirodnog jezika, obucavanje, zakljucivanje, pretraga, vidjenje
- Afektivno racunarstvo je grana VI koja se avi projektovanjem sistema i uredjaja koji mogu respoznavati, interpretirati i obradjivati emocije
- Automatizacija je kada se izvrsavaju precizne akcije koje se ponavljaju i unapred su programirane
- Autonomija je kada se akcija generise i izvrsava radi ostvarivanja cilja kada izvrsavanje moze biti ometano okruzenjem sto zahteva od sistema da se dinamicki adaptira i da vrsi ponovno planiranje
- Autonomija je kada se moze opazati, delovati i rasudjivati, a ne samo izvrsavati ponovne stvari kao kod automatizacije
- Tri osnovne robotske reci: opazanje, plan i delovanje

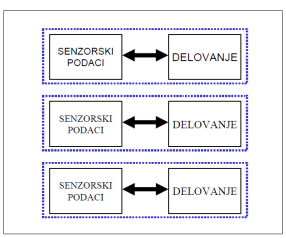
Paradigme:

1. Hijerarhijska paradigma je monolitna i spora:
2. Reaktivna paradigma je kada nema plana i ponasanja su nezavisna i izvrsavaju se u paraleli
3. Hibridna paradigma je kada se desi prvo planiranje pa potom ponasanje dok se ne zavrsi ili dok se ne sagleda potreba da se promeni

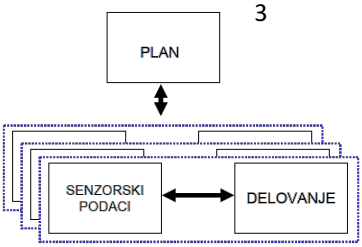
1



2



3



Neuralne mreze

- **Generalizacija** je sposobnost produkovanja zadovoljavajuceg izlaza i za ulaze koji nisu bili prisutni u toku obucavanja
- Generalizacija je omogucena osobinom da se slicni ulazi slikaju u slicne izlaze
- **Obucavanje** je proces menjanja sinaptickih tezina u cilju dostizanja zeljenih performansi mreze
- Podsecaju na ljudski mozak u dva pogleda: uce procesom obucavanja, tezine veza sluze za memorisanje znanja
- **Adaptivnost** je sposobnost menjanja jacine veza
- **Evidencionalni odziv** znaci da mreza kao izlaz moze da produkuje i stepen uverenja o datoj odluci
- **Kontekstualna informacija** znaci da je svaki neuron u mrezi pod uticajem globalne aktivnosti ostalih neurona
- Neuralne mreze su fizicki sistemi koji mogu prikupljati, uksladjivati i koristiti eksperimentalno znanje
- **Neuralna mreza** je zbir vestackih neurona koji su medjusobno povezani i interaktivni kroz operacije obrade signala
- Svaki neuron sadrzi: nivo aktivnosti, izlaznu vrednost, skup ulaznih veza, vrednost pomeraja i skup izlaznih veza
- Tok ulaznih i izlaznih signala je jednosmeran
- **Perceptron** (jednoslojna mreza) znaci da je ulaz neurona linearna kombinazija ulaznih veza uvecana za vrednost pomeraja koji je nezavisan od ulaza
- Ako se perceptron koristi za klasifikaciju, linearna kombinacija se provlaci kroz step funkciju
- Neuralna mreza bez skrivenih jedinica ima jednak lokalni i globalni minimum
- Neuralna mreza sa skrivenim jedinicama ima vise lokalnih minimuma pa postoji mogucnost da se zavrsi u nekom od njih
- **Univerzalna aproskimativnost**: mreze mogu da aproksimiraju sve kontinualne funkcije
- **Hebovo pravilo**: Kada je akson ćelije A veoma blizu toga da pobudi ćeliju B i ponavljajući i uporno pokušava da je aktivira, dešava se određeni proces rasta ili metaboličke promene u jednoj ili u obe ćelije, čiji rezultat je porast efikasnosti ćelije A, kao jedne od ćelija koje pobuđuju ćeliju B
- Tezine se menjaju kada se zeljeni i trenutni izlazi razlikuju

- Koraci obucavanja:
1. Bira se brzina obucavanja
2. Inicijalizuju se tezina kao male slucajne vrednosti
3. Dovodjenje ulaznih vrednosti
4. Azuriranje tezina

- Osnovna ideja delta pravila je da se definise mera performanse sistema i onda da se optimizuje ta performansa

- Koraci delta obucavanja:
1. Bira se brzina obucavanja i maksimalna dozvoljena greska za kraj obucavanja
2. Inicijalizuju se tezina kao male slucajne vrednosti
3. Dovodjenje ulaznih vrednosti
4. Izracunava se ukupna greska mreze
5. Azuriranje tezina
6. Ako se nije iskoristio ceo skup obucavanja, vraca se na korak 3
7. Ciklus je zavrsen. Ako je trenutna greska mreze veca o maksimalne dozvoljene, krece novi ciklus.

- Podaci obucavanje ne sadrze informciju o tome koje su vrednosti izlaza na cvorovima skrivenog sloja pa treba izvrsiti povratno prositranje greske sa izlaznog slona na skrivene slojeve
- Pravilo po kome se menjaju tezine u obucavanju povratnim prostiranjem greske da to je metodom opadanja gradijenta

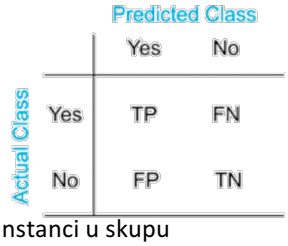
- Koraci obucavanja propagacijom greske unazad:
1. Bira se brzina obucavanja i maksimalna dozvoljena greska za kraj obucavanja
2. Inicijalizuju se tezina kao male slucajne vrednosti
3. Dovodjenje ulaznih vrednosti
4. Izracunava se ukupna greska mreze
5. Azuriranje tezina izlaznog neurona
6. Azuriranje tezina skrivenog neurona
7. Ako se nije iskoristio ceo skup obucavanja, vraca se na korak 3
8. Ciklus je zavrsen. Ako je trenutna greska mreze veca o maksimalne dozvoljene, krece novi ciklus

Masinsko ucenje

- Masinsko ucenje je sposobnost softvera da generalizuje na osnovu prethodnog iskustva i da koristi kreirane generalizacije kako bi pruzio odgovore na pitanja sa kojima nije imao iskustva

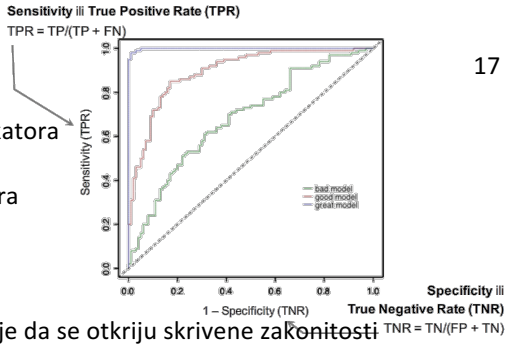
- Osnovni oblici:

1. Nadgledano ucenje - u skupu za obucavanje se dobijaju ulazi i izlazi, a zadatak je da na neobelezenom ulazu dodeli tacni izlaz
    Klasifikacija - skup izlaznih vrednosti je diskretan - zadatak je odredjivanje klase kojoj neka instanca pripada
    - zavisno od broja klasa razlikujemo binarnu klasifikaciju (2 klase) i vise-klasnu klasifikaciju
    - kod viseklasne klasifikacije algoritam u svakoj iteraciji nauci da odvoji jednu klasu od ostalih
    - Algoritmi:
        - Neuralne Mreze (NN)
        - Logistic Regression
        - Decision Trees
            * Ideja je da se podeli prostor atributa kojima su objekti opisani u vise razlicitih i medjusobno neprekloplјenih regiona
            * Klasa novog objekta ce biti dominantna klasa u regionu u kojem se nalazi i koji je odredjen pomocu njegovih atributa
            * Podela prostora atributa je iterativni proces koji se sastoji od izbora atributa koji ce biti osnova za podelu i izbora vrednosti atributa koja ce posluziti kao granicna vrednost
            * Treba naci regione tako da se minimizuje greska pri klasifikaciji
            * Rekurzivna binarna podela prostora atributa je pristup koji se primenjuje da bi se identifikovali regioni koji minimizuju gresku pri klasifikaciji i njegove osobine su:
                Top down pristup - krece se od vrha stabla gde sve instance pripadaju jednom regionu i zatim se prostor deli na regione
                - **Greedy pristup** - pri svakom koraku najbolja podale se odredjuje na osnovu stanja u tom koraku, tj. ne uzima se u obzir sta ce biti u narednim koracima koji mogu dovesti do manje greske
            * Osim greske pri klasifikaciji, cesto se koriste Gini index i Cross entropy koji predstavljaju cistocu regiona tako sto ti regioni imaju visok procenat instanci koji pripadaju istoj klasi
            * Orezivanje stabla (tree pruining) je resenje za problem velikih overfitting stabala
                - Preporuka je da se primenom cross validacije utvrdi greska pri klasifikaciji za podstabla razlicite velicine i izabrati podstablo koje daje najmanju gresku
            * Prednosti stabala odlucivanja:
                - Mogu se graficki prikazati
                - Mogu se koristiti za klasifikaciju i regresiju
                - Mogu se koristiti i u slucaju da atributi imaju nedostajuce vrednosti
            * Nedostaci stabala odlucivanja:
                - Daju slabije rezultate nego druge metode nadgledanog ucenja
        - Support Vector Machines (SVM)
        - K Nearest Neighbors (KNN)
        - Naive Bayes
    - Mere uspesnosti klasifikatora:
        - Matrica zabune (Confusion Matrix)



TP = True Positive
FP = False Positive
TN = True Negative
FN = False Negative

        - Tacnost (Accuracy)
            * Procenat instanci koji su uspesno klasifikovani
            * Accuracy = (TP + TN) / N, gde je N ukupan broj instanci u skupu
            * U slucaju neravnomerne raspodele instanci izmedji klasa (skewed classes) ova mera je nepouzdana
        - Preciznost (Precision) i odziv (Recall)
            * Precision = TP / (TP + FP), od svih poruka koje su oznacene kao spam, koji procenat su stvarno spam
            * Recall = TP / (TP + FN), od svih poruka koje su stvarno spam, koji procenat je oznacen kao spam
            * Precision i Recall su obrnuto proporcionalne
        - F mera (F measure)
            * Kombinuje preciznost i odziv i omogucuje jednostavnije poredjenje algoritama
            * $F = (1 + ß^2) * Precision * Recall / (ß^2 * Precision + Recall)$
            * Gde ß kontrolise koliki znacaj ima odziv
            * F1 mera daje jednak znacaj preciznosti i odzivu



- Povrsina ispod ROC krive
    * Primenjuje se za merenje performansi binarnih klasifikatora
    * Vrednost je u intervalu 0-1
    * Ako je vrednost 0.5, znaci da je metoda slucajnog izbora
    * Sto je vrednost veca od 0.5, to je klasifikator bolji

Regresija - skup izlaznih vrednosti je kontinualan

2. Nenadgledano ucenje - u skupu za obucavanje se dobijaju samo ulazi, a zadatak je da se otkriju skrivene zakonitosti
    - **Klasterizacija** - zadatak grupisanja instanci po slicnosti koja se procenjuje pomocu Euklidske ili Manhattan distance
        * Ne postoji tacno resenje pa samo domenski eksperti mogu da ocene uspesnost
        * K means algoritam radi tako sto za definisan broj klastera slucajno postavlja centroide cije se pozicije menjaju iterativno kako bi udaljenost instanci od centroida bila minimalna, tj. funkcija kostanja (funkcija distorzije)
        * Zavisno od inicijalnog izbora centroida, algoritam moze konvergirati brze ili sporije
        * Visestruka nasumicna inicijalizacija omogucava da se izbegnu lokalni minimumi tako sto vise puta nasumicno postavlja centroide, izvrsava aloritam i izracunava funkciju kostanja pa izabere najbolje pozicije za centroide
        * Ako posedujemo znanje o pojavi koju podaci opisuju, mozemo pretpostaviti broj klastera, inace krenemo od malog broja klastera i u vise iteracija testiramo model uvek sa jednim klasterom vise
        * Kriterijumi za procenu kvaliteta klastera:
            1. **Medjusobna udaljenost tezista**: sto su tezista dalja, to je kvalitet veci
            2. **Standardna devijacija**: sto ima manje instanci koje su dalje od grupe, to je kvalitet veci
            3. **Suma kvadrata**: suma kvadrata odstupanja instanci u okviru klastera od tezista klastera

3. Ucenje sa podsticajem - agent deluje na okruzenje izvrsavanjem niza akcija, a posle okruzenje deluje na agenta u vidu kazni i nagrada i cilj agenta je da vremenom maksimizuje nagrade

- Osnovni koraci pocesa masinskog ucenja:
1. Prikupljanje podataka
2. Priprema podataka
3. Analiza i unapredjivanje skupova podataka
4. Izbor jednog ili vise metoda masinskog ucenja
5. Obuka i evaluacija izabranih modela
6. Izbor modela za koriscenje

- Odabir metode masinskog ucenja zavisi od:
1. Vrste problema koji se resava
2. Karakteristika skupa atributa (tip, homogenost, kolinearnost)
3. Obim podataka koji su na rasponaganju

- Podaci se pomocu slucajne selekcije dele na 60% za trening, 20% za validaciju i 20% za testiranje
- Podaci za trening i validaciju se koriste za poboljsavanje jednog modela (validacija se koristi da se izbegne overfitting)
- Podaci za testiranje se koriste da potvrde tacnost modela i za poredjenje vise modela
- **Unakrsna validacija (cross validation)** se koristi za efikasno koriscenje podataka i funkcionise tako sto skup podataka za trening podeli na K delova, zatim se obavlja K iteracija treninga i validacije modela i u svakoj iteraciji se uzima 1 deo za potrebe validacije, a ostalih K-1 delova se koristi za treniranje; pri svakoj iteraaciji se racunaju performanse modela i na kraju se racuna prosecna uspesnost na nivou K iteracija
- **Analiza greske** podrazumeva rucno pregledanje primera na kojima je model pravio greske i uocavanje skrivenih zakonitosti
- **Atributi** omogucavaju da entiteti budu medjusobno razliciti i izazov je naci te atribute
- **Overfitting** se odnosi na situaciju u kojoj model savrseno nauci da vrsi predikciju za instance iz trening skupa, ali ima veoma slabu sposobnost predikcije za nevidjene instance; usko je vezan za varijansu
- **Varijansa** ukazuje na to u kojoj meri bi se kreirani model promenio ukoliko bi doslo do promene podataka u koriscenom skupu za trening; sto je metoda slozenija, to se njena varijansa biti veca
- **Underfitting** se odnosi na situaciju ukojoj model ne uspeva da aproksimira podatke za treing tako da ima slabe performanse cak i na trening skupu; usko je vezan za bias
- **Bias** se odnosi na gresku koja se javlja u slucaju koriscenja vrlo jednostavnog modela za potrebe resavanja slozenog realnog problema; sto je metoda slozenija, to ce bias biti manji