SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Primjena genetskog programiranja u razvoju umjetne inteligencije

Željko Rumenjak

Voditelj: *Doc.dr.sc. Domagoj Jakobović*

Zagreb, svibanj, 2007.

Sadržaj:

1. Uvod

Genetsko programiranje je relativno mlada tehnika koja brzo sazrijeva i približava se konkretnim primjenama u raznim djelatnostima. Genetsko programiranje omogućuje računalu da riješi problem, a da ga nitko nije eksplicitno programirao za rješavanje tog problema. To se postiže stvaranjem populacije računalnih programa koji će se sami unaprjeđivati kako bi što bolje riješili zadani problem. Nadalje, genetsko programiranje je već u nekoliko slučajeva postiglo rezultate koji se mogu mjeriti sa onima postignutim od ljudi.

Ta obilježja čine genetsko programiranje posebno zanimljivom metodom za ostvarivanje umjetne inteligencije. Za uspješno ostvarivanje umjetne inteligencije program mora moći ispitati veliki broj mogućnosti koje može učiniti na svakom koraku, a upravo to genetsko programiranje jako dobro radi. Još jedna velika prednost genetskog programiranja je što ono radi sa cijelim populacijama programa pa si tako može dopustiti i ispitivanje nekih mogućnosti koje se u početku ne čine dobrima, a na kraju mogu dati bolje rezultate.

Pomoću genetskog programiranja moguće je napraviti računalo koje će, ovisno o situaciji u kojoj se nalazi, generirati program koji najbolje rješava tu situaciju i izvršiti ga. Najveća prepreka za takvu primjenu genetskog programiranja u umjetnoj inteligenciji je njegova potreba za računalnom snagom. Genetski programi zahtijevaju brza računala i zbog toga se za umjetnu inteligenciju ne koriste programi koji bi se sami mogli prilagođavati svakoj situaciji, već se prvo generira program koji će zadovoljavati određene zahtjeve i taj se program nakon toga više ne može mijenjati.

Bez obzira na te nedostatke (koji će se možda razvojem tehnologije zaobići) primjena genetskog programiranja za rješavanje problema povezanih s umjetnom inteligencijom može donijeti veliki napredak u njenom razvoju. Možda najjači argument za to je da ono koristi principe evolucije, a to je jedini proces za koji sigurno znamo da je proizveo inteligenciju.

2. Umjetna inteligencija

Termin umjetna inteligencija prvi je uveo John McCarthy 1956. godine, kad je rekao da se inteligencija može opisati tako detaljno da je moguće napraviti stroj koji će je oponašati. Definirao je umjetnu inteligenciju kao znanstvene i inženjerske djelatnosti izrade inteligentnih strojeva, odnosno računalnih programa. Umjetna inteligencija je snažno povezana s računalnom znanošću, ali ima i važne veze s matematikom, psihologijom, biologijom i mnogim drugima.

2.1. Jaka i slaba umjetna inteligencija

Umjetna inteligencija podijeljena je u dva razreda: jaka umjetna inteligencija i slaba umjetna inteligencija. Jaka umjetna inteligencija tvrdi da je moguće napraviti računala koja će misliti bar na razini na kojoj razmišljaju ljudi. Ta računala imaju i znaju koristiti znanje koje obuhvaća široko područje i posjeduju određeni stupanj samosvjesnosti.

Slaba umjetna inteligencija tvrdi da se računalima mogu dodati samo neka obilježja slična razmišljanju kako bi bili korisniji alati, da strojevi mogu samo simulirati ljudski način razmišljanja, odnosno djelovati kao da su inteligentna. Računala koja posjeduju slabu inteligenciju ne mogu biti samosvjesna niti mogu posjedovati širok raspon kognitivnih sposobnosti koje imaju ljudi. Računala sa takvim obilježjima su se već počela primjenjivati, na primjer za prepoznavanje govora.

2.2. Turingov test

Turingov test je eksperiment kojeg je predložio matematičar Alan Turing u njegovom članku „Computing Machinery and Intelligence“ iz 1950. godine. On tvrdi kako bi se računalo trebalo smatrati inteligentnim ako može uspješno uvjeriti drugog čovjeka da je i ono samo čovjek. U Turingovom testu sudac razgovara (preko tipkovnice) sa dva sustava, jedan je čovjek, a drugi računalo. Razgovor može biti o bilo čemu i traje određeno vrijeme koje je unaprijed određeno. Turing tvrdi da ako sudac na kraju razgovora ne može razlikovati računalo od čovjeka na temelju razgovora, moramo reći da je računalo inteligentno.

Neki smatraju da Turingov test ne može biti točna definicija inteligentnog stroja zbog bar slijedeća tri razloga:

1. Stroj koji je prošao Turingov test može samo oponašati ljudske navike u razgovoru, ali to može biti puno slabiji uvjet od inteligencije. Stroj može samo pratiti neka pomno osmišljena pravila. Česti protuargument tome je pitanje: „Kako znamo da ljudi samo ne prate neka pomno osmišljena pravila?“

2. Stroj može biti inteligentan, a da ne zna razgovarati sa ljudima.

3. Mnogi ljudi koje bi vjerojatno smatrali inteligentnima možda ne bi prošli Turingov test (na primjer mala djeca ili nepismeni ljudi). S druge strane inteligencija ljudi se gotovo uvijek mjeri na temelju govora.

Loebnerova nagrada se svake godine dodjeljuje programu za koji suci smatraju da je najsličniji čovjeku od onih koji su sudjelovali na natjecanju. Prva i druga nagrada još nikad nisu bile dodijeljene.

2.3. Grane umjetne inteligencije

Neke od grana umjetne inteligencije su:

- Logička umjetna inteligencija: Sve što program zna o svijetu, što treba napraviti u određenoj situaciji i njegovi ciljevi opisano je matematičkim jezikom logike. Program u svakoj situaciji odabire one akcije za koje zaključi da su primjerene ostvarivanju njegova cilja.

- Prepoznavanje uzoraka: Obično kada program detektira nešto novo, on to uspoređuje sa nekim uzorkom. Na primjer program za prepoznavanje slike će na slici pokušati pronaći uzorak očiju i nosa kako bi pronašao lice. Proučavaju se i kompliciraniji uzorci, na primjer uzorci za prepoznavanje teksta.

- Zaključivanje na temelju postojećeg znanja: Iz nekih činjenica mogu se zaključivanjem dobiti druge. Za neke primjene je dovoljna matematička dedukcija, ali od 1970-ih programima su dodavane i neke nove metode zaključivanja. Najjednostavniji primjer takvih metoda je podrazumijevano zaključivanje u kojem se donosi podrazumijevani zaključak, ali taj zaključak se može povući ako se pojave činjenice koje tvrde suprotno.

- Znanje i zaključivanje temeljeno na općem znanju: To je područje u kojem je umjetna inteligencija najdalje od ljudske iako je to bilo aktivno područje istraživanja od 1950-ih godina. Postignut je velik napredak, no potrebno je još mnogo novih ideja kako bi se to područje stvarno razvilo.

- Učenje na temelju iskustva: Pristupi umjetnoj inteligenciji temeljeni na neuronskim mrežama se specijaliziraju za to područje. Strojevi mogu učiti samo iz činjenica ili ponašanja koje njihov formalizam može protumačiti, na žalost gotovo svi sustavi učenja imaju jako ograničene sposobnosti razumijevanja informacija.

- Planiranje: Programi za planiranje počinju od općenitih činjenica o svojoj okolini (posebno utjecaja pojedinih akcija na okolinu), činjenica o trenutnoj situaciji i njihovog cilja. Oni od tih činjenica generiraju strategiju za ostvarivanje cilja.

- Genetsko programiranje: Genetsko programiranje je tehnika kojom se razvijaju programi za rješavanje problema tako da se križaju slučajno odabrani programi iz populacije i odabiru oni koji su najuspješniji u rješavanju zadanog problema.

3. Genetsko programiranje

3.1. Evolucija u prirodi

Genetsko programiranje se temelji na principima evolucije. Evolucija se u prirodi događa kad su zadovoljena slijedeća četiri uvjeta:

1. Organizmi se mogu razmnožavati
2. Nekoliko organizama tvori populaciju
3. Postoje razlike između organizama u populaciji
4. Neki organizmi iz populacije mogu preživjeti u okolini bolje nego drugi

Evolucija djeluje nad cijelom populacijom, prirodni odabir djeluje nad jedinkama. Jedinke koje mogu preživjeti u okolini bolje od ostalih će se razmnožavati i tako povećati broj jedinki poput njih u populaciji. Sposobnost jedinke da preživi u okolini se obično zove dobrota (eng. *fitness*). U procesu evolucije strukturu populacije određuju dobrote njezinih jedinki.

3.2. Genetski algoritmi

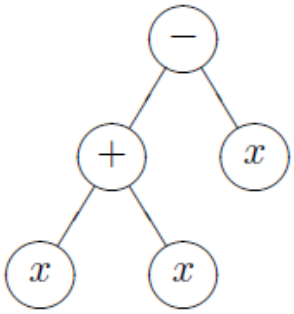
Za uspješnu primjenu genetskog programiranja potrebno je poznavanje genetskih algoritama. Genetski algoritmi počeli su se razvijati 1960-ih kao reakcija na *top-down* pristup koji je koristila većina ljudi koji su se bavili istraživanjima vezanim za umjetnu inteligenciju u to vrijeme.

John Holland sa University of Michigan smatrao je da je potreban drugačiji pristup za rješavanje problema iz domene umjetne inteligencije. U to vrijeme pokušavalo se definirati sva moguća pravila i uvjete za bilo koji događaj koji se je mogao pojaviti. Holland je smatrao da to nije dobar pristup jer ako se dogodi nešto za što ne postoji pravilo, cijeli program se u većini slučajeva srušio. Upravo zbog toga se rješenja temeljena na umjetnoj inteligenciji nisu mogla koristiti u većini djelatnosti (uvijek se može dogoditi nešto nepredviđeno).

Holland je odlučio da je najbolji način da se napravi sustav temeljen na umjetnoj inteligenciji *bottom-up* odnosno induktivni pristup. Holland je uveo genetske algoritme kao način na koji se može simulirati prirodna evolucija u umjetnim sustavima. Osmislio je sustav nizova bitova sastavljenih od nula i jedinica, umjesto 4 nukleotida koji se nalaze u molekulama DNA. Kada se definiraju veličina populacije i duljina nizova bitova, nizovi se popune slučajno odabranim kombinacijama nula i jedinica, odrede se vrijednosti funkcije dobrote za svaki niz i odvija se proces reprodukcije. Nad nizovima mogu djelovati i drugi genetski operatori kao što su križanje i rekombinacija. Većina operacija koje se odvijaju nad nizovima bitova se mogu odvijati paralelno. Na taj način genetski algoritmi iskorištavaju paralelizam i druge jedinstvene karakteristike koje ih čine jako korisnom i moćnom tehnikom za rješavanje različitih problema.

3.3. Genetsko programiranje

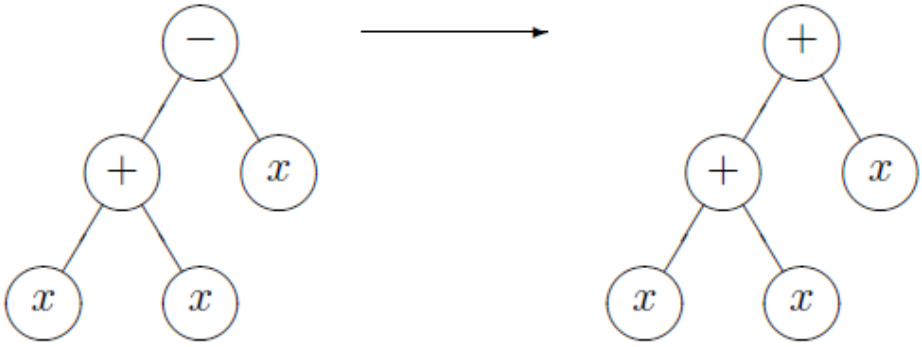
John Koza je proširio genetske algoritme tako da rade s računalnim programima umjesto sa nizovima bitova i nazvao to genetskim programiranjem. Ti programi se tipično prikazuju kao stabla. Na slici 3.1 prikazan je primjer jednog programa prikazanog u obliku stabla.



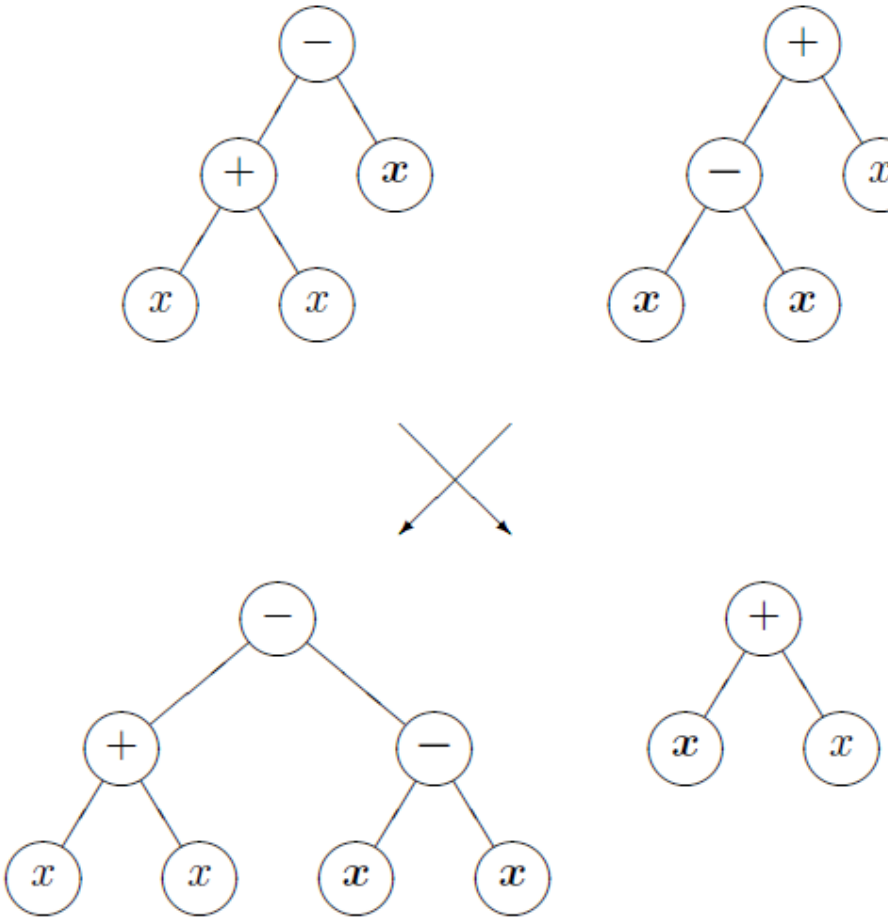
Slika 3.1: Prikaz jednostavnog programa koji obavlja funkciju (x + x) - x

Glavni cilj genetskog programiranja je rješavanje problema tako da između svih mogućih programa koji rješavaju zadani problem pronađe program sa što većom vrijednošću funkcije dobrote. Programi su sastavljeni od dijelova koda koji odgovaraju tipu zadanog problema, a početna populacija se jednostavno dobiva slučajnim spajanjem tih dijelova koda.

Za svaki program se određuje dobrota, isto kao i za nizove bitova kod genetskih algoritama. Oni sa najvećom vrijednošću funkcije dobrote se razmnožavaju, dok oni sa najmanjom vrijednošću izumiru. Programi generirani u prvoj generaciji gotovo uvijek imaju jako malu vrijednost funkcije dobrote za netrivijalne probleme. Bez obzira na to, neki programi iz populacije će imati veću vrijednost funkcije dobrote od drugih. Te razlike se koriste kako bi se ostatak pretrage usmjerio prema obećavajućim područjima. Programi iz populacije koji će sudjelovati u različitim genetskim operacijama se odabiru ovisno o vrijednostima funkcije dobrote. Mali dio populacije (oko 9%) se kopira iz jedne generacije u drugu. Jako mali dio populacije (oko 1%) se mutira (slučajno se odabiru dijelovi programa koji se mijenjaju). Najveći dio populacije sudjeluje u genetskoj operaciji razmnožavanja u kojoj se stvaraju programi tako da se rekombiniraju dijelovi dvaju programa roditelja.



Slika 3.2: Genetska operacija mutacije. Lijeva jedinka mutira u desnu kada se u njezinom korijenu znak - zamjenjuje sa znakom +.



Slika 3.3: Prikaz genetske operacije razmnožavanja

Stvaranje početne populacije i stvaranje programa pomoću genetskih operacija se obavljaju na način da rezultiraju sintaksno ispravnim programima. Nakon što se nad trenutnom generacijom obave genetske operacije i stvori se nova generacija, tada ta nova generacija zamjenjuje trenutnu. Postupci određivanja vrijednosti funkcije dobrote, selekcije i obavljanja genetskih operacija se ponavljaju kroz mnogo generacija.

U svakoj slijedećoj generaciji pojavljuju se programi sa sve većom vrijednošću funkcije dobrote, odnosno programi koji sve bolje rješavaju problem jer se populacija generira na principima evolucije i prirodnog odabira. Eventualno će se pronaći i program koji rješava zadani problem dovoljno dobro. Ako se, iz nekog razloga, problem (okolina) promijeni, promijenit će se i funkcija dobrote i počet će se pojavljivati novi programi koji rješavaju taj problem. Takva sposobnost prilagodbe je velika prednost genetskog programiranja.

Kad su jedinke u populaciji računalni programi (umjesto, na primjer, nizova bitova), Hollandov genetski algoritam može se upotrijebiti za pronalaženje programa koji rješava određeni problem. Genetsko programiranje se može koristiti za rješavanje mnogo različitih tipova problema, a može se koristiti i za postizanje rješavanja problema od strane računala bez prethodnog programiranja računala za rješavanje tog problema.

3.4. Nelogična priroda kreativnosti i evolucije

Biološka pozadina koja leži iza genetskih algoritama i genetskog programiranja razlikuje te metode od svih drugih koje su se prije pokušale iskoristiti za automatsku izradu računalnih programa. Mnogi računalni znanstvenici i matematičari smatraju da svaka tehnika koja služi za rješavanje bilo kakvog problema mora biti deterministička i da se mora oslanjati na logiku. U skladu s tim većina konvencionalnih metoda za ostvarivanje umjetne inteligencije i strojnog učenja konstruirane su tako da imaju te karakteristike. Međutim logika ne upravlja sa dva najvažnija procesa za rješavanje složenih problema: procesom otkrivanja novih načina za rješavanje problema, odnosno izuma i procesom evolucije koji se odvija u prirodi. Nova ideja koja se može logički izvesti iz poznatih činjenica, korištenjem transformacija koje su dostupne i poznate ne smatra se izumom. Kako bi nešto bilo izum, prema zakonu za patente, mora postojati „nelogični korak“. Logično razmišljanje je sigurno korisno za mnoge primjene, ali nije dovoljno za proces izumljivanja.

Genetsko programiranje u svojoj potrazi za optimalnim programom koji rješava zadani problem nije opterećeno predrasudama koje obično tjeraju ljude na slično razmišljanje niti metodama koje nameće formalna logika. Zbog toga se upravo genetskim programiranjem često mogu dobiti novi i neočekivani načini rješavanja problema, koji mogu biti bolji od onih osmišljenih od ljudi.

Kada je cilj automatsko stvaranje računalnih programa smatra se da je, ne nužno logičan, pristup prisutan u procesima evolucije i otkrivanja izuma puno učinkovitiji od onih vođenih logikom koje koriste konvencionalna umjetna inteligencija i strojno učenje.

4. Genetsko programiranje: Turingov treći način za ostvarivanje umjetne inteligencije

Jedan od glavnih izazova u računalnoj znanosti je postići da računalo može rješavati probleme za čije ga rješavanje nije nitko eksplicitno programirao. Posebno lijepo bilo bi imati sustav koji je neovisan o problemu, čiji bi ulaz bio samo opis zadanog problema, a izlaz program koji taj problem rješava. Izazov je postići da računalo može napraviti ono što se od njega traži bez da mu je itko morao opisati kako to točno treba napraviti.

Alan Turing je prepoznao da je umjetnu inteligenciju moguće ostvariti pristupom temeljenim na biologiji. U svojem eseju "Intelligent Machines" iz 1948. godine Turing je povezao tehnike pretraživanja i izazov da se napravi računalo koje će rješavati probleme bez da ga se mora eksplicitno programirati. Zatim je naveo tri načina na koja se tehnike pretraživanja mogu koristiti kako bi se napravio inteligentan računalni program.

Prvi način koji je Turing naveo bila je pretraga koja se temeljila na logici. Taj način reflektira Turingov rad na logičkoj osnovi računalnih algoritama.

Drugi način Turing je nazvao „kulturološka pretraga“. Taj način se oslanja na stručno znanje skupljeno od drugih tijekom perioda od više godina i sličan je današnjim sustavima koji se temelje na pretraživanju znanja.

Treći način koji je Turing naveo je „genetska ili evolucijska pretraga“. Kod tog načina pretražuju se kombinacije gena, a kriterij je mogućnost preživljavanja. Veliki uspjeh tih pretraga na neki način potvrđuje ideju da se intelektualna aktivnost uglavnom sastoji od različitih načina pretrage.

Turing je predložio kako se evolucija i prirodna selekcija mogu uključiti u proces traženja inteligentnih strojeva. Rekao je da se ne može očekivati da ćemo pronaći dobar inteligentni stroj u prvom pokušaju, nego da je potrebno pokušati učiti taj stroj i vidjeti kako dobro uči. Nakon toga može se isto pokušati sa drugim i vidjeti da li je drugi bolji ili gori od prvoga.

Od 1940-ih potrošeno je mnogo truda pokušavajući ostvariti umjetnu inteligenciju pomoću prvog i drugog Turingovog načina. Nasuprot tome Turingovom trećem načinu (genetsko i evolucijsko pretraživanje) posvećeno je puno manje vremena.

Smatra se da je razumno očekivati da sustav koji će automatski generirati računalne programe treba imati većinu ili sva od sljedećih obilježja[3]:

1. Počinje od onoga što treba napraviti: sustav počinje od opisa problema koji je potrebno riješiti
2. Odgovara na pitanje „Kako riješiti problem?“: daje rezultat u obliku koraka koje je potrebno obaviti kako bi se problem riješio
3. Vraća računalni program: rezultat je program koji se može izvesti na računalu
4. Određivanje veličine rješenja, odnosno programa: ima sposobnost određivanja broja koraka koje je potrebno izvršiti kako bi se došlo do rješenja i ne zahtjeva od korisnika da unese točnu veličinu rješenja

5. Ponovno korištenje koda: zna automatski organizirati korisne korake u grupe kako bi se oni kasnije mogli lakše ponovno koristiti
6. Parametrizirano ponovno korištenje koda: može ponovno koristiti grupe koraka s drugim vrijednostima varijabli
7. Korištenje memorije: zna koristiti memoriju za pojedinačne varijable, vektore, matrice, polja, stogove, redove, liste i druge strukture podataka
8. Iteracije, petlje i rekurzija: zna koristiti iteracije, petlje i rekurziju
9. Hijerarhijska organizacija: zna hijerarhijski organizirati grupe koraka
10. Automatsko određivanje strukture programa: zna automatski odrediti da li da koristi potprograme, petlje, rekurziju ...
11. Široko područje programskih konstrukcija: zna implementirati i koristiti pokazivače, uvjetne operatore, funkcije za rad sa različitim tipovima podataka...
12. Dobro definiran: radi na dobro definiran način. Točno zna što korisnik mora unijeti i što sustav mora vratiti.
13. Neovisan o problemu: korisnik ne mora mijenjati sustav za svaki novi problem
14. Široka uporaba: daje zadovoljavajuća rješenja za različite probleme iz različitih područja
15. Dobro rješava veću inačicu istog problema
16. Može se mjeriti sa rezultatima postignutim od ljudi: daje rezultate koji se mogu mjeriti sa onima koje daju ljudi koji se bave područjem iz kojeg je problem

Genetsko programiranje trenutno bezuvjetno posjeduje 13 od 16 obilježja za koja je razumno očekivati kako bi svaki sustav koji automatski generira računalne programe trebao imati i barem djelomično posjeduje preostala tri[3].

Svojstvo pod rednim brojem 16 je posebno važno jer govori da je konačan cilj sustav koji proizvodi korisne programe, a ne samo programe koji rješavaju jednostavne probleme.

U „Genetic Programming III: Darwinian Invention and Problem Solving“ navodi se 14 slučajeva gdje je genetsko programiranje proizvelo program koji se može mjeriti sa rezultatima postignutim od ljudi[3].

4.1. Umjetna inteligencija koja se može mjeriti sa ljudskom

Kad se kaže da se umjetno stvoreno rješenje može mjeriti sa ljudskim, ne misli se na vrijeme koje mu treba da ispiše 10000 stranica ili da može izračunati π na milijun decimala. Misli se na to da je način na koji je računalo riješilo problem bolji od onog na koji su ga rješavali ljudi, ne zbog brzine niti točnosti na koju računalo može nešto izračunati, već zbog toga što je ono otkrilo novu tehniku za rješavanje zadanog problema.

Naravno ne tvrdi se da je genetsko programiranje jedini mogući pristup izazovu da računala mogu rješavati probleme bez da ih se eksplicitno programira za njihovo rješavanje. Međutim danas ne postoji niti jedna druga metoda za ostvarivanje umjetne inteligencije koja posjeduje više od samo nekoliko od 16 nabrojanih obilježja.

Zanimljivo je da niti jedno od 16 obilježja ne zahtjeva da metoda za ostvarivanje sustava za automatsko generiranje računalnih programa mora biti temeljena na formalnoj logici ili imati eksplicitnu bazu znanja. Tijekom prošla 4 desetljeća vladalo je vjerovanje da je

moguće postići da računala automatski rješavaju probleme samo primjenom formalne logike i znanja. Taj pristup tipično obuhvaća odabir reprezentacije znanja, prikupljanje znanja, kodiranje znanja u bazu znanja, pohranjivanje baze znanja u računalo i manipulaciju znanja u računalu pomoću metoda formalne logike.

Međutim postojanje opće prihvaćenog mišljenja kroz 4 desetljeća ne znači, samo po sebi, da je mišljenje i točno. Popularnost mišljenja ne znači da ne postoji alternativni način za postizanje određenog cilja. Posebno, popularnost prvog (temeljenog na logici) i drugog (temeljenog na znanju) Turingovog načina tijekom prošla 4 desetljeća, ne znači da treći Turingov način za ostvarivanje umjetne inteligencije ne može biti uspješniji.

Upravo to potvrđuju rezultati koje genetsko programiranje postiže. Postoji 36 slučajeva gdje je genetsko programiranje postiglo rezultate koji se mogu mjeriti sa onima postignutim od ljudi. Od tih 36 slučajeva u njih 15 postignuta je funkcionalnost koja odgovara patentiranom izumu iz 20. stoljeća, u 6 slučajeva isto je postignuto za izum iz 21. stoljeća, a u 2 slučaja rezultat je bio novi izum koji se može patentirati.

4.2. Razlike između genetskog programiranja i drugih pristupa umjetnoj inteligenciji

Genetsko programiranje se razlikuje od svih drugih pristupa ostvarivanju umjetne inteligencije u svima (ili većini) od slijedećih sedam obilježja:

1. Reprezentacija
2. Uloga transformacija točke u točku (eng. *point-to-point*) u pretrazi
3. Uloga algoritma penjanja na brdo (eng. *hill climbing*) u pretrazi
4. Uloga determinizma u pretrazi
5. Uloga eksplicitne baze znanja
6. Uloga metoda formalne logike u pretrazi
7. Pozadina tehnike

Prva razlika između genetskog programiranja i drugih metoda se odnosi na način na koji se reprezentiraju podaci. Većina tehnika umjetne inteligencije koristi specijalizirane strukture podataka. Osim u rijetkim situacijama, programeri ne koriste takve strukture podataka. Umjesto toga oni već desetljećima pišu programe koji koriste različite operacije (aritmetičke i logičke) nad različitim tipovima podataka (integer, floating-point, boolean, ...). Programeri koriste internu memoriju računala za spremanje rezultata raznih operacija kako se oni ne bi morali svaki put ponovno izračunavati kad su potrebni, koriste petlje i rekurzije. Organiziraju korisne dijelove koda u grupe (potprogrami) kako bi ih mogli kasnije jednostavnije ponovno koristiti i organiziraju te potprograme u hijerarhije. Sve nabrojene postupke programeri su koristili od pojave prvih računala u 1940-ima. Međutim bez obzira na dokazanu korisnost tih metoda, one su jako rijetko prisutne kod postojećih tehnika umjetne inteligencije, a i kad jesu prisutne, toliko su izmijenjene da gotovo nisu prepoznatljive. Suprotno od toga genetsko programiranje koristi većinu tih metoda.

Druga razlika tiče se prirode pretrage. Skoro sve ostale metode jednu točku u području pretrage transformiraju u neku drugu točku. Genetsko programiranje je drugačije jer se kod njega razvija populacija različitih jedinki koje često predstavljaju kontradiktorne

pristupe rješavanju zadanog problema. Genetsko programiranje zbog toga ne transformira jedan program u drugi, nego transformira cijelu populaciju programa u drugu populaciju programa.

Treća razlika je u pristupu penjanja na brdo. Kada se put kroz područje pretrage kreće od jedne točke do druge točke, uvijek postoji gotovo neodoljiva želja da se za sljedeću točku odabere ona za koju se zna da je bolja od drugih mogućih točaka. Zbog toga se gotovo sve tehnike pretraživanja oslanjaju na pohlepni algoritam penjanja na brdo (eng. *greedy hill climbing)* kako bi napravile transformaciju iz trenutne točke u sljedeću. Ta želja se još više pojačava jer se mnogi jednostavni problemi iz područja umjetne inteligencije i strojnog učenja mogu dobro riješiti na taj način. No popularnost ne može ukloniti problem zaustavljanja pretrage u lokalnom optimumu koji nije i globalni optimum. Zanimljivo je da gotovo svi problemi koji nisu trivijalni imaju optimalne točke koje nisu dostupne s pohlepnim algoritmom penjanja na brdo. Dosta dobra definicija problema koji nije trivijalan je da postoje točke nedostupne pohlepnom penjanju na brdo. Činjenica da genetsko programiranje transformira početni skup točaka u skup drugih točaka pomaže mu pri rješavanju tog problema. Genetsko programiranje može alocirati određeni broj putova prema točkama za koje je poznato da su inferiorne. Upravo takve točke često znaju otkriti put koji će na kraju dovesti do boljeg rješenja. Činjenica da genetsko programiranje djeluje nad cijelim populacijama omogućuje mu da napravi određen broj „avanturističkih“ poteza dok istovremeno prati putove koji trenutno daju veće nagrade. Naravno genetsko programiranje nije jedina tehnika pretraživanja koje izbjegava zastajanje u lokalnom optimumu, ali većina tehnika koje se danas primjenjuju u polju umjetne inteligencije ostaje zarobljena u lokalnom optimumu.

Četvrta razlika je u tome da genetsko programiranje obavlja pretragu koja se temelji na vjerojatnosti. Ponovno genetsko programiranje nije jedinstveno u tome, ali većina tehnika za pretraživanje koje se danas koriste u polju umjetne inteligencije je deterministička.

Peto, u genetskom programiranju većinom nije uključena eksplicitna baza znanja. Iako postoji mnogo načina na koji se znanje može uključiti u genetsko programiranje, ono ne zahtjeva (niti obično koristi) eksplicitnu bazu znanja za upravljanje njegovom pretragom.

Šesto, genetsko programiranje se ne oslanja na metode formalne logike za upravljanje svoje pretrage. Procesima evolucije i prirodne selekcije ne vlada logika, često se pojavljuju nekonzistentnosti i kontradiktorne alternative. Rješenja koja su razvijena evolucijom se gotovo uvijek razlikuju od onih razvijenih konvencionalnim metodama umjetne inteligencije. Rješenja dobivena evolucijom se obično mogu puno bolje nositi sa stalnim promjenama koje se događaju u stvarnoj okolini.

Sedmo, biološka metafora koja se nalazi u temeljima genetskog programiranja se jako razlikuje od temelja svih drugih tehnika koje se koriste za automatsko generiranje računalnih programa. Mnogi znanstvenici iz računalne znanosti i matematičari su zbunjeni prijedlogom kako bi biologija mogla biti važna za njihovo polje. Baš suprotno, biologiju ne treba gledati kao nešto što nema veze sa područjem umjetne inteligencije, već kao izvor iz kojeg treba uzimati ideje. Uistinu, genetsko programiranje se temelji na jedinoj metodi koja je ikad proizvela inteligenciju – metodi evolucije i prirodne selekcije.

5. Primjene

5.1. Timski rad pomoću genetskog programiranja

U ovom projektu rješava se problem sakupljanja hrane pomoću genetskog programiranja. Genetsko programiranje će razviti programe koji oponašaju mrave koji sakupljaju hranu. Postoje dva posebna faktora zbog kojih mravi moraju raditi zajedno, umjesto da svaki radi sam.

Prvo, u okolišu postoji rijeka koju mravi moraju prijeći kako bi došli do hrane. Iako svaki mrav izvršava isti program, neki moraju ući u vodu i poginuti kako bi napravili most preko kojega drugi mogu proći. Isto tako mravi koji prežive moraju znati da je već sagrađen most koji mogu koristiti, umjesto da poginu pokušavajući napraviti drugi most.

Drugo, hrana će biti preteška da bi je jedan mrav sam mogao podići. Mravi moraju pronaći hranu i na neki način pozvati druge mrave da im pomognu. Mravi komuniciraju feromonima. Ako svi mravi čekaju na pomoć kod nakupina hrane, neki, ali ne svi, moraju shvatiti da se dogodio potpuni zastoj i otići od svoje hrane kako bi pomogli drugim mravima.

Oba ta problema zahtijevaju da mravi koriste timski rad kako bi sakupili svu hranu. Mravi moraju shvatiti što drugi mravi rade bez direktne međusobne komunikacije.

Postoje 4 vrste testova na kojima se rješenje ispitivalo:

1. Jednostavan problem sakupljanja hrane. U tom testu mravi su slučajno tražili hranu po mapi, nosili je natrag do gnijezda i ostavljali trag feromona do hrpe hrane iza sebe.

2. Drugi problem je uključivao i rijeku. Mravi su odlično uspijevali prijeći vodu, a napravili su i oblak feromona oko hrane prije nego su krenuli natrag prema gnijezdu. Dobiveno rješenje je najbolje radilo sa mnogo mrava. Kad mrav uđe u vodu i umre, blizu njega mora biti drugi mrav koji će primijetiti oblak feromona koji je on proizveo prije nego umre. Kada ima puno mrava (oko 100) tada postoji dosta dobra šansa da će drugi mrav biti blizu i otprilike 10 mrava od 100 će možda umrijeti. Ako se rješenje pokrene sa 30 mrava, tada će vjerojatno svi umrijeti jer niti jedan neće primijetiti oblak feromona od drugih zbog toga što su predaleko.

3. U problemu sa teškom hranom (jedan mrav je ne može sam odnijeti) sa mogućnošću potpunog zastoja dobiveno rješenje se temeljilo na sreći. Problem je bilo lagano riješiti ako se ukloni mogućnost potpunog zastoja. S mogućnošću potpunog zastoja genetsko programiranje je pronašlo točno rješenje koje se temeljilo na sreći jer nije naišlo na potpuni zastoj. Genetsko programiranje je na kraju pronašlo rješenje koje radi, ali to nije dovoljno u stvarnom svijetu jer dobiveno rješenje radi samo u malom broju situacija. Genetsko programiranje je razvilo rješenje koje je radilo u svakom slučaju ako se koristio veći broj mrava, ali tada mravi uopće nisu koristili timski rad.

4. Zadnji problem je bio kombinacija problema sa prijelazom preko vode (2.) i problema sa teškom hranom (3.). Dio problema sa prijelazom preko vode nije bilo teško riješiti, a dio problema sa teškom hranom je imao iste poteškoće kao i 3. slučaj. Genetsko programiranje je pronašlo savršeno rješenje, ali ono se temeljilo na sreći i nije uvijek radilo.

Ako je problem pretežak, tada dobiveno rješenje možda neće raditi za svaku situaciju. U ostalim situacijama genetsko programiranje će pronaći rješenje koje će uvijek raditi. Kad genetsko programiranje razvije rješenje za jedan problem, nije sigurno da će se to rješenje moći primijeniti za neki drugi problem koliko god on bio sličan početnome.

5.2. PADO – Parallel Algorithm Discovery and Orchestration

Sustavi umjetne inteligencije trebaju razumjeti mnogo informacija iz vanjskog svijeta, ali oni iz svijeta primaju veliku količinu informacija koje sadrže puno smetnji. Za uspješan daljnji razvoj umjetne inteligencije potrebno je osigurati dobar sustav za razumijevanje tih signala.

PADO je zamišljen kao sustav za razumijevanje i pretvorbu bilo kakvih signala, a za domenu njegova testiranja odabrani su vizualni signali. Upravo ti signali su odabrani zato jer je rješenje takvih problema najpotrebnije za daljnji razvoj umjetne inteligencije. Ljudi najviše ovise o pretvorbi vizualnih signala i ta pretvorba se u mozgu odvija gotovo bez ikakvog truda. Nadalje strojno učenje je imalo vrlo malo uspjeha u tom području.

PADO je tehnika koja uči algoritme za prepoznavanje signala direktno, tako da ne postoji ugrađen način na koji algoritam pretražuje sliku i dolazi do zaključka. Za postizanje takvog učenja PADO koristi strategije evolucije.

PADO ne ovisi o tipu signala kojeg treba klasificirati i stoga obećava slične rezultate za signale različitih tipova, od sonara, preko govora pa do teksta.

Napravljeni su problemi za testiranje koji su namjerno teži od onih koje prepoznavanje objekata obično zahtjeva. Na objektima su rađene translacije, rotacije, mijenjalo se osvjetljenje i njihove dimenzije. Uz to objekti koji su se koristili za testiranje nisu bili jednostavni, jednobojni niti geometrijski pravilni.

Na tim teškim problemima PADO je postigao razinu prepoznavanja objekata od 50%, a ta razina je porasla na 95% kada su se koristile slike sa originalnim bojama.

Te rezultate PADO je postigao bez ikakve pomoći od korisnika bez gotovo ikakvog znanja iz tog područja. U PADO su bile implementirane samo najjednostavnije funkcije za obradu vizualnih signala (Pixel, Region, Least, Most, Average, and Variance). Činjenica da su te funkcije bile napisane za pola sata i da su bile dovoljne za prepoznavanje zadanih slika podupire hipotezu da PADO može raditi sa jako malo ili bez bilo kakve vanjske intervencije.

Dizajn sustava PADO pruža nekoliko zanimljivih mogućnosti:

1. U bilo kojem trenutku tijekom procesa učenja program, kao skup pravila za razumijevanje signala, se može izdvojiti iz sustava i odmah koristiti za prepoznavanje objekata.

2. PADO u svojem dizajnu koristi metode evolucije i zbog toga može koristiti mnogo različitih pristupa rješavanju problema.

3. Korištenje neovisnih rješenja omogućuje izvođenje PADO sustava na paralelnim računalima za postizanje gotovo linearnog ubrzanja.

4. Neovisnost arhitekture o vrsti signala koji se ispituje omogućuje korištenje PADO sustava za bilo koji tip signala.

Istraživači današnjeg doba potroše puno vremena na traženje pravog algoritma za rješavanje njihovog problema i na prilagođavanje tog algoritma dok ne počne davati zadovoljavajuće rezultate. To ne dovodi do pravog napretka jer jednostavno postoji previše načina na koje se to može napraviti. PADO može istovremeno pratiti mnogo različitih metoda za rješavanje problema i upravo zbog toga PADO ima velike šanse da uvelike pridonese daljnjem razvoju sve boljih rješenja za razumijevanje signala iz prirode. Njegovi tvorci se nadaju da će uskoro postići puno veću uspješnost prepoznavanja vizualnih signala i da će se ta uspješnost održati i na signalima drugih tipova.

Slika 5.1: Neke slike koje su se koristile za testiranje sustava PADO. Lijeva slika je bila pokazana sustavu za učenje, a desnu je sustav trebao prepoznati.

6. Zaključak

Postoji nekoliko faktora koji najbolje opisuju genetsko programiranje kao uspješnu metodu za ostvarivanje umjetne inteligencije:

- Genetskim programiranjem danas se ostvaruje umjetna inteligencija koja se može mjeriti s ljudskom.
- Genetsko programiranje je automatiziran proces za stvaranje izuma.
- Genetsko programiranje stvara parametrizirani graf koji je opće rješenje danog problema.
- Genetsko programiranje je počelo iskorištavati prednosti dobivene stalnim razvojem i povećanjem brzine računala.

Jedan od faktora koji najviše ograničavaju genetsko programiranje je brzina računala. Povećanje u brzini znači da populacije mogu biti mnogo veće što znači da će genetsko programiranje moći provesti mnogo širu pretragu u prostoru računalnih programa. Ostvarivanje paralelizma koji se događa u evoluciji u prirodi je možda ključ za ostvarivanje boljih algoritama genetskog programiranja.

Ako će povećanja u brzini računala biti popraćena i učinkovitijim algoritmima pretrage genetskog programiranja, moguće je da smo već započeli povijesni prijelaz iz ere pisanja računalnih programa prema eri gdje ćemo moći reći računalu točno kako da uči i kako da samo programira sebe i druga računala.

7. Literatura

[1] Molnar D., Genetic Programming, Will Bill Gates become Billy Appleseed?, http://www.genetic-programming.com/published/computerbits0696.html, 15.3.2007.

[2] BBC The Next Big Thing – Artificial Intelligence, http://www.open2.net/nextbigthing/ai/ai_in_depth/in_depth.htm, 21.4.2007.

[3] Koza J. R., Bennett F. H. III i dr., Genetic Programming: Turing's Third Way to Achieve Machine Intelligence, 29.7.1999., http://citeseer.ist.psu.edu/cache/papers/cs/8064/http:zSzzSzwww.genetic-programming.comzSzEUROGEN99TURING.pdf/koza99genetic.pdf, 18.3.2007.

[4] Koza J.R., Human-Competitive Machine Intelligence by Means of Genetic Algorithms, http://www.cs.umt.edu/u/wright/evolcomp/miniproj/koza1999prog_computers.pdf, 15.3.2007.

[5] Teller A., Veloso M., PADO: Learning Tree Structured Algorithms for Orchestration into an Object Recognition System, 10.2.1995., http://citeseer.ist.psu.edu/cache/papers/cs/6364/ftp:zSzzSzreports.adm.cs.cmu.eduzSz1995zSzCMU-CS-95-101.pdf/teller95pado.pdf, 15.3.2007.

[6] LaLena M., Teamwork in Genetic Programming, magistarski rad, Rochester Institute of Technology School of Computer Science and Technology, 1997

8. Sažetak

Ovaj seminarski rad navodi neke razloge za primjenu genetskog programiranja na probleme vezane za umjetnu inteligenciju. Genetsko programiranje koristi metode procesa evolucije i prirodnog odabira i zbog toga procesi koji se odvijaju prilikom stvaranja i razvoja programa nisu nužno logični, smatra se da upravo to omogućuje da genetsko programiranje proizvede neočekivane rezultate koji mogu biti bolji od onih koje postižu ljudi.

Na početku rada navode se osnovne informacije o umjetnoj inteligenciji i ukratko se objašnjavaju neke grane umjetne inteligencije. Zatim slijedi definicija i opis rada genetskog programiranja i ono se uspoređuje sa 2 važna procesa u prirodi: procesom pronalaska izuma i procesom evolucije. U glavnom dijelu ukratko se objašnjavaju tri Turingova načina za ostvarivanje umjetne inteligencije i navode se obilježja koja bi trebao imati sustav koji automatski generira računalne programe. Na kraju su objašnjene neke od primjena genetskog programiranja na probleme povezane s umjetnom inteligencijom. Opisuje se ostvarivanje timskog rada pomoću genetskog programiranja u problemu sakupljanja hrane i projekt PADO. U projektu PADO genetsko programiranje se koristi za otkrivanje algoritma za prepoznavanje signala iz vanjskog svijeta. Za sada se PADO koristi samo za video signale, ali trebao bi raditi jednako dobro sa bilo kojom vrstom signala.