SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Kartezijsko genetsko programiranje

Iva Brajer

Voditelj: doc.dr.sc. Domagoj Jakobović

Zagreb, travanj, 2010

Sadržaj

Uvod

Genetsko programiranje, u svojoj originalnoj formi, svodi se na evoluciju programa u obliku LISP stabala parsiranja. Stablo je poseban oblik grafa u kojem dva čvora uvijek imaju između sebe najmanje jedan put[1]. Glavna motivacija uvođenja stabla kao reprezentaciju rješenja bila je omogućiti promjenjivu veličinu genotipa.

Međutim, postoji i pristup pod nazivom kartezijsko genetsko programiranje (eng. *Cartesian Genetic Programming*; skraćeno *CGP*) gdje je genotip prikazan kao niz stalne veličine sastavljen od cjelobrojnih vrijednosti koje se preslikavaju u usmjerene grafove umjesto u stabla. Jedna od motivacija za takvu reprezentaciju rješenja bila je omogućavanje općenitijeg prikaza rješenja jer je struktura stabla podskup skupa usmjerenih grafova.

Kartezijsko genetsko programiranje pokazalo se iznimno učinkovitim pri učenju logičkih funkcija, u toj mjeri da je bilo bolje od genetskog i evolucijskog programiranja.

Postavlja se pitanje može li se upotreba kartezijskog genetskog programiranja proširiti i na ostala područja i hoće li se njegova upotreba također pokazati učinkovitijom od „starijih“ metoda.

U radu se razmatra što je točno kartezijsko genetsko programiranje te pojmovi vezani uz njega, definicija kartezijskog programa, stvaranje kartezijskog genotipa, zatim razlike između njega i genetskog programiranja (s osvrtom na mane i prednosti), gdje se ovaj genotip primjenjuje i je li se uistinu pokazao isplativijim.

Također, bit će ukratko objašnjena implementacija kartezijskog genotipa u sklopu *Evolutionary Computation Frameworka*.

[1] Put je sekvenca spojenih čvorova.

1. Kartezijski genotip

Kartezijsko genetsko programiranje nosi takav naziv iz razloga što se grafička interpretacija njegovog fenotipa može predstaviti kao dvodimenzionalna mreža čvorova preslikana u kartezijski koordinatni sustav (vidi sliku Slika 1) [1].

Genotip je predstavljen nizom cjelobrojnih vrijednosti i stalne je duljine. Inicijalni ulazi[2] i izlazi iz čvorova se slijedno numeriraju. Funkcije koje se stavljaju u čvorove[3] se također slijedno numeriraju. Čitav genotip je samo niz ulaza u čvorove i funkcija s dodanim izlazima na kraju genotipa. Korisno je zamijetiti kako izlazi iz čvorova nisu dio genotipa niti je to potrebno jer se slijedno numeriraju po čvorovima (vidi slika Slika 1) [1]. Također, u ovom prikazu genotipa svaki čvor ima isti broj ulaza i jedan izlaz. To je opće prihvaćena konvencija (koja će se ovdje pridržavati), ali ne mora biti tako i čvorovi u genotipu mogu imati varijabilan broj ulaza i izlaza [2].

Genotip se na određen način preslikava u fenotip koji zapravo predstavlja usmjeren graf. Premda je genotip stalne duljine, fenotip je promjenjive duljine ovisno o broju neizraženih gena[4] [1].

Važnost preslikavanja genotipa u fenotip je činjenica da to omogućava da se više genotipa preslika u isti fenotip i time je eksplicitno zadovoljena neutralnost. Neutralnost (eng. *neutrality*) označava prisutnost različitih genotipova s istom vrijednosti dobrote (eng. *fitness*) [1].

U kartezijskom genetskom programiranju zapravo postoji velik broj genotipova koji se preslikavaju u isti fenotip, a to je moguće zahvaljujući velikoj količini zalihosti (eng. *redundancy*). Postoje tri vrste zalihosti [1]:

- zalihost čvorova (eng. *node redundancy*) - nastaje zbog gena povezanih s čvorovima koji nisu dio usmjerenog grafa (ako graf predstavlja program)
- funkcionalna zalihost (eng. *functional redundancy*) - nastaje zbog određenog broja čvorova koji zajednički tvore podfunkciju koja se zapravo može prikazati s puno manjim brojem čvorova
- zalihost ulaza (eng. *input redundancy*) - nastaje ako neki od inicijalnih ulaza nisu povezani s nijednim ulazom u čvorove

Zalihost čvorova i zalihost ulaza zajednički tvore potencijal za dodavanje korisne neutralnosti jer nepovezani čvorovi mogu proći kroz niz neutralnih promjena i kasnije u procesu evolucije postati povezani s ostatkom grafa. Time se u konačnici postiže dobivanje veće vrijednosti dobrote.

Funkcionalna zalihost bi se s druge strane trebala izbjegavati jer uzrokuje nepotrebno povećavanje genotipa (a što je veći genotip to je i prostor pretraživanja rješenja veći).

---

[2] Inicijalni ulazi su zapravo skup terminala iz genetskog programiranja.
[3] Funkcijski čvorovi su funkcijski skup iz genetskog programiranja.
[4] Pod neizraženim genima misli se na nespojene izlaze nekih čvorova što označava nepovezan čvor u grafu npr. sivi kvadrati na slici Slika 1.

Ako postoji više genotipova koji u populaciji imaju najbolju vrijednost dobrote tada se za najbolji genotip bira neki od njih slučajnim odabirom. To se podrazumijeva pod pojmom neutralne pretrage (eng. *neutral search*) [1].

2. Kartezijski program

Kartezijski program (eng. *Cartesian program*; skraćeno *CP*) koji se označava s *P* je definiran kao skup [1]

$$P = \{G, n_i, n_o, n_n, F, n_f, n_r, n_c, l\}$$

gdje je

- G - genotip kao niz cjelobrojnih vrijednosti gdje je
  - n_i - broj ulaza u program
  - n_n - broj ulaza (ulaznih veza) u čvor (eng. *node input connections*)
  - n_o - broj izlaza iz programa
- F - skup funkcija u čvorovima gdje je
  - n_f - broj mogućih funkcija
- n_r - broj redaka
- n_c - broj stupaca
- l - parametar stupnja povratka (eng. *levels back parameter*) koji određuje koliko će prijašnjih stupaca čvorova moći spojiti svoje izlaze na čvor u trenutnom stupcu[5]

Kartezijsko genetsko programiranje u svojoj najjednostavnijoj inačici razmatra samo povezanost čvorova s jednosmjernim prolazom unaprijed (eng. *feed forward connectivity*), međutim u njegov genotip se mogu lako uključiti i petlje, ali ti slučajevi se neće razmatrati.

Inicijalni ulazi programa i izlazi iz čvorova se smiju povezati s bilo kojim ulazom u čvorove u idućem stupcu, odnosno stupcima (ovisno o parametru stupnja povratka). Čvorovi u istom stupcu ne smiju biti međusobno spojeni jedni s drugim, ali zato bilo koji čvor može biti ili povezan ili nepovezan [1].

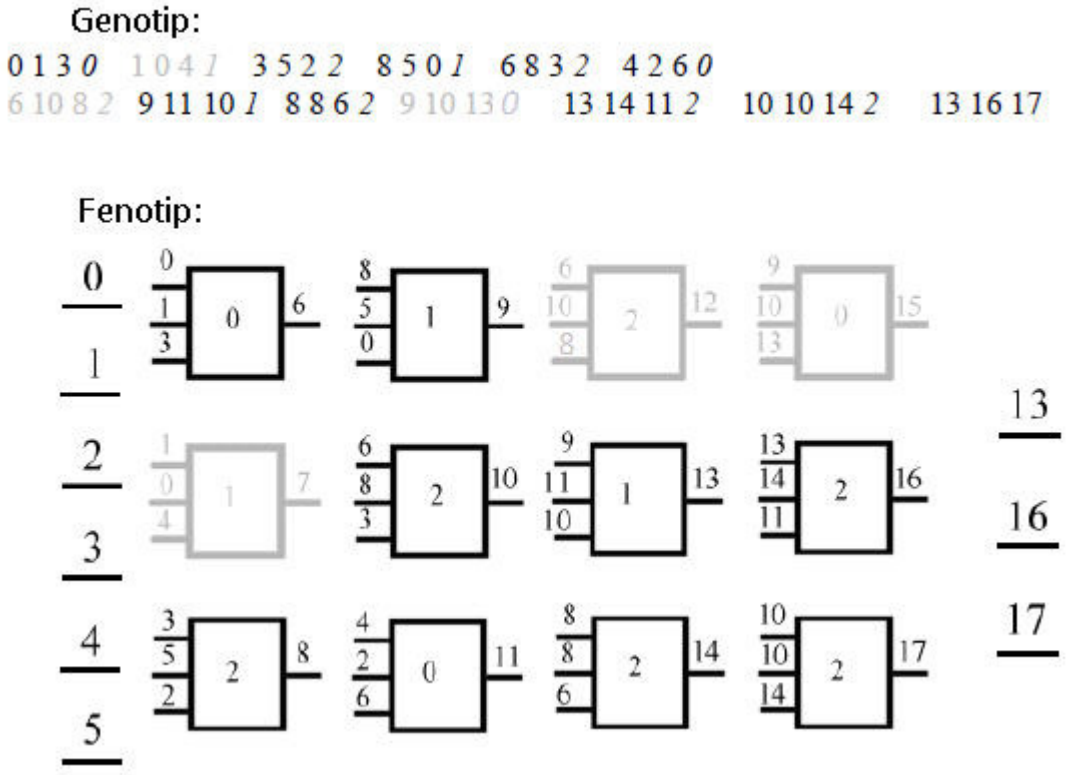
Primjer kartezijskog programa je na slici Slika 1. Uzete su vrijednosti:

- $n_i = 6$ - ulazi su označeni redom 0-5
- $n_n = 3$ - npr. za čvor koji se nalazi u prvom retku i prvom stupcu ulazne veze su 0, 1 i 3
- $n_o = 3$ - izlazi su redom 13, 16 i 17
- $n_f = 3$ - funkcije su redom numerirane kao 0, 1 i 2, dok u stvarnosti to mogu biti zbrajanje, oduzimanje i množenje
- $n_r = 3$ - tri su retka čvorova
- $n_c = 4$ - četiri su stupca čvorova
- $l = 2$ - moguće je dohvatiti izlaze čvorova udaljene za najviše dva stupca unazad da ih se spoji na ulaz čvora u trenutnom stupcu (npr. ulazi čvorova u

[5] Pritom se na primarne ulaze odnosi isto kao i na izlaze čvorova.

zadnjem stupcu se zbog toga nikako ne mogu spojiti sa izlazima čvorova u prvom stupcu)

Dužina genotipa je stalna i iznosi $n_r \cdot n_c \cdot (n_n + 1) + n_o$. Međutim, to je samo maksimalna dužina kartezijskog programa koja jest stalna dok stvarna veličina (gledano kao usmjereni graf) može biti varijabilna, ali ograničena maksimumom. Na primjer, na slici Slika 1 je samo 9 čvorova povezano, nepovezani čvorovi su označeni sivom bojom i označavaju čvorove s kojima se nitko od čvorova u narednim stupcima nije povezao [1].



Slika 1. Preslikavanje genotipa u fenotip za program sa šest ulaza, tri izlaza i tri funkcije (0, 1, 2 u kvadratima, naznačene kurzivom u genotipu). Sivi kvadrati označavaju nepovezane čvorove. [1]

Pri inicijalizaciji genotipa ili primjenom nekog od operatora mutacije, promijenjeni geni se moraju pridržavati određenih ograničenja (pravila) koja osiguravaju da je genotip valjan, odnosno da je preslikavanje genotipa u fenotip smisleno [1].

Neka c_{kj}^{n} označava gen (jedan od n_n ulaznih veza u čvor) koji je k-ti ulaz (ulazna veza) u čvor u j-tom stupcu[6]. Pritom se treba pridržavati pravila [1]

$$e_{min} = n_i + (j - l) \cdot n_r$$

$$e_{max} = n_i + j \cdot n_r$$

$$c_{kj}^{n} < e_{max} \quad , \qquad ako\ j < l$$

[6] Prvi najljeviji stupac je označen kao nulti stupac.

$$e_{min} \leq c_{kj}^{n} < e_{max} \quad , \qquad ako\ j \geq l$$

Neka c_k^o označava gen (jedan od n_o izlaza) koji je k-ti izlaz iz programa. Pritom se treba pridržavati pravila [1]

$$h_{min} = n_i + (n_c - l) \cdot n_r$$
$$h_{max} = n_i + n_c \cdot n_r$$
$$h_{min} \leq c_k^o < h_{max}$$

Neka c_k^f označava gen (jedan od n_f funkcija) koji predstavlja funkciju u k-tom čvoru. Pritom treba vrijediti [1]

$$0 \leq c_k^f < n_f$$

Ukoliko se pridržava ovih pravila, križanje se može slobodno primijeniti tako da daje valjano rješenje.

Moderne verzije kartezijskog genetskog programiranja osim jednostavnih funkcija koje se ugrađuju u čvorove (npr. osnovne operacije, trigonometrijske, logičke) dozvoljavaju i upotrebu potprograma kao funkcija što se zapravo može shvatiti kao manji genotip unutar većeg. U genetskom programiranju je to poznato kao automatski definirane funkcije (eng. *Automatically Defined Functions*; skraćeno *ADF*), dok se u kartezijskom genetskom programiranju više spominju automatski iskoristivi izlazi (eng. *Automatic Re-used Outputs*; skraćeno *ARO*) premda ne funkcioniraju na isti način kao *ADF* [1]. Promatrat će se samo jednostavne funkcije ugrađene u čvorove.

3. Usporedba s genetskim programiranjem (prednosti i mane)

Kartezijsko genetsko programiranje najviše je uspjeha pokazalo pri učenju logičkih funkcija (korisno u problemu automatskog oblikovanja sklopovlja) jer pruža veliku količinu lokalne pretrage za razliku od genetskog programiranja [2].

Genetsko programiranje koristi varijabilnu veličinu genotipa za razliku od kartezijskog što omogućuje puno veću ekspresivnost prostora rješenja i globalniju pretragu. Međutim, to često ne znači i veće mogućnosti da rješenje bude nađeno jer veći prostor rješenja podrazumijeva i više potrebnog vremena za pronalaženje optimalnog rješenja. Također, zbog učestalih primjena križanja i mutacije, primjetan je i nagli rast genotipa tokom evolucije (eng. *bloat*) [3].

Kartezijsko genetsko programiranje je ograničeno u pretrazi prostora rješenja sa stalnom duljinom genotipa, ali je time izbjegnut nagli rast genotipa. S druge strane, stalna duljina genotipa omogućuje lokalno pretraživanje dok je kod genetskog programiranja pretraga globalna [1].

Kao što je možda već uočeno, u genetskom programiranju ne postoji fenotip nego samo genotip iz čega se zaključuje da ne postoji neutralna pretraga.

Prikaz kartezijskog genotipa je neovisan o tipu podataka i pri evaluaciji dobrote nije potrebno unutar genotipa pamtiti podatke kao što se to radi u genetskom

programiranju gdje se terminali zamjenjuju za konstante, a unutar funkcijskih čvorova upisuje međurezultat nastao propagacijom vrijednosti od njihove djece [2].

Križanje u kartezijskom genetskom programiranju je smisleno i funkcionalno, a mutacija vrlo jednostavna (pod uvjetom da se pridržavaju predefinirana ograničenja). Genetsko programiranje jest pri primjeni mutacije u prednosti jer se ne treba paziti na predefinirana pravila nego samo da struktura stabla parsiranja ostane sačuvana.

Također, za čvorove vrijedi da ne moraju nužno biti povezani za razliku od genetskog programiranja [2].

Mnogo je prirodnije implementirati potprograme (*ADF* u *GP*, *ARO* u *CGP*) u kartezijskom genotipu zbog intuitivnijeg prikaza [2].

Pokazano je da je u kartezijskom programiranju dovoljna manja populacija kako bi se došlo do optimalnog rješenja za razliku od genetskog programiranja koje zahtjeva veliku populaciju [2].

Osim učenja logičkih funkcija, kartezijsko genetsko programiranje se iskušalo i u radu s realnim vrijednostima i vremenski ovisnom ponašanju te se nije pokazalo lošijim u odnosu na genetsko programiranje, ali ne i pretjerano boljim [1].

Također, pokušalo se pomoću kartezijskog genetskog programiranja riješiti neke standardne probleme koji se inače rješavaju pomoću genetskih algoritama (npr. *onemax* problem). Kartezijsko genetsko programiranje se pokazalo boljim od klasičnih metoda genetskih algoritama, pretpostavlja se upravo zbog mogućnosti neutralne pretrage i prisutnosti zalihosti [4].

Kartezijsko genetsko programiranje je učinkovitije za evoluciju manjih programa i pokazuje se općenito efikasnijim od genetskog programiranja ako je očekivano rješenje jednostavno, dok je s druge strane evolucija većih programa ipak prepuštena genetskom programiranju [3].

4. Implementacija u *ECF*-u

Kartezijski genotip je implementiran u okviru *Evolutionary Computation Frameworka* (*ECF*) [5] [6]. Zamišljen je po definiciji kao niz cjelobrojnih vrijednosti stalne duljine, čvorovi su povezani tako da je ostvaren jednosmjeran prolaz unaprijed te se pridržavalo pravila i konvencije objašnjene u prvom i drugom dijelu seminara.

Od funkcijskih čvorova implementirane su zbrajanje, oduzimanje, množenje, dijeljenje te sinus i kosinus.

Implementirana funkcija dobrote radi evaluaciju genotipa prema principu jednosmjeran prolaz unaprijed zamjenjujući elemente iz domene za inicijalne ulaze koji predstavljaju varijable. Inicijalni ulazi mogu biti i predefinirane konstante, ali one se onda ne mijenjaju s elementima iz domene nego se direktno prosljeđuju dalje po genotipu. Na kraju se dobiveni izlazi uspoređuju s elementima kodomene.

Inicijalizacija genotipa se odvija slučajnim odabirom vrijednosti pri čemu se genotip puni cjelobrojnim vrijednostima koje predstavljaju ulaznu vezu u čvor, funkciju ili izlaz pritom se pridržavajući unaprijed definiranih pravila za njihov odabir.

Operator križanja je križanje s jednom točkom prijeloma pri čemu slučajno odabrana točka prijeloma može biti bilo koja cjelobrojna vrijednost iz genotipa. Dijete se dobiva spajanjem lijevog dijela jednog roditelja s desnim dijelom drugog roditelja s obzirom na točku prijeloma.

Operator mutacije je mutacija s slučajno odabranom jednom točkom promjene pri čemu se treba pridržavati predefiniranih ograničenja prilikom slučajno odabrane vrijednosti koja dolazi na mjesto stare vrijednosti.

Zaključak

Predstavljeno je kartezijsko genetsko programiranje, noviji oblik genetskog programiranja, koje koristi jednostavniji prikaz genotipa i na neki način popunjava nedostatke uočene u genetskom programiranju.

Iznimno je korisno u problemima učenja logičkih funkcija i automatskog oblikovanja sklopovlja jer pokazuje veću pristranost za lokalnom pretragom nego genetsko programiranje. U velikoj mjeri se koristi neutralnom pretragom i zalihošću koji se smatraju korisnim svojstvima za povećanje dobrote genotipa.

Međutim, osim ograničene veličine genotipa, upitan je i velik broj parametara koji se treba odrediti prije rješavanja problema (broj inicijalnih ulaza, ulaza u čvorove, izlaza, funkcija itd.). Za razliku od genetskog programiranja gdje je dovoljno odrediti maksimalan rast stabla u dubinu te skup primitiva, kartezijsko genetsko programiranje puno više ovisi o odabiru optimalnih parametara.

Upitno je koliko je u ostalim problemima odabir kartezijskog genetskog programiranja bolji od genetskog programiranja i može li se sa sigurnošću tvrditi da je kartezijsko genetsko programiranje generalno gledano bolje. Zasada je jedino potvrđeno da je za očekivana manja rješenja kartezijski genotip bolji odabir. Međutim, ne preostaje ništa drugo nego isprobati kartezijsko genetsko programiranje i za ostale probleme, usporediti rezultate s drugim evolucijskim metodama i eventualno graditi poboljšanja.

Literatura

[1] Miller, J. F., & Thomson, P. (2000). Cartesian Genetic Programming. *Proceedings of the Third European Conference on Genetic Programming (EuroGP2000)* (str. 121-132). Berlin: Springer-Verlag.

[2] Miller, J. F. (1999). An empirical study of the efficiency of learning Boolean functions using a Cartesian Genetic Programming Approach. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)* (str. 1135-1142). San Francisco: Banzhaf, Wolfgang.

[3] Lones, M. (20. travanj 2004). *Genetic Programming*. Preuzeto 14. travanj 2010 iz User Web Areas at University of York: http://www-users.york.ac.uk/~mal503/common/thesis/c6.html

[4] Walker, J. A., & Miller, J. F. (2007). Changing the Genospace: Solving GA Problems with Cartesian Genetic Programming. *Proceedings of the 10th European conference on Genetic programming (EuroGP2007)* (str. 261-270). Berlin: Springer-Verlag.

[5] Jakobović, D. (23. veljača 2010). *Evolutionary Computation Framework*. Preuzeto 25. veljača 2010 iz ZEMRIS: http://gp.zemris.fer.hr/ecf/

[6] Brajer, I. (24. travanj 2010). Preuzeto 25. travanj 2010 iz svn://smaug.zemris.fer.hr/ecf/branches/CGP

Sažetak

Uvod u kartezijsko genetsko programiranje, pojam preslikavanja genotipa u fenotip, pojam neutralnosti i zalihosti. Ostvarenje kartezijskog programa i kartezijskog genotipa. Pravila za održavanje ispravnosti kartezijskog genotipa tokom procesa evolucije. Usporedba kartezijskog genetskog programiranja s genetskim programiranjem s osvrtom na mane i prednosti. Primjene kartezijskog genetskog programiranja. Kratak opis implementacije kartezijskog genotipa u *Evolutionary Computation Frameworku*.