SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1902

Određivanje pravila trgovanja dionicama uz pomoć genetskog programiranja

Aleksandar Topuzović

Zagreb, rujan 2011.

SADRŽAJ

1. Uvod

Trgovanje na tržištima kapitala unosan je posao. Kao takav u zadnje vrijeme sve više i više pobuđuje interes javnosti. Tako su interes za trgovanje vrijednosnim papirima, poglavito dionicama, pokazali i građani Republike Hrvatske pogotovo nakon inicijalnih javnih ponuda[1] INA-e 2006 godine i HT-a 2007 godine. Sljedeća 2008 godina bila je godina u kojoj se najviše trgovalo. Većina malih investitora na tržištu nema strategiju ulaganja, dok se pak oni veći redovito služe raznim analizama kao pomoć pri odlukama o kupovini. Najčešća metoda analize je tehnička analiza koja barata povijesnim cijenama i obimima trgovanja. Svrha ovog rada je istražiti na koji način koristeći genetsko programiranje generirati pravila trgovanja na burzi.

U poglavlju 2 opisano je što su to tržišta kapitala, te da li je moguće predvidjeti cijene i kojim metodama. Opisani su i neki od standardnih indikatora tehničke analize te na koji se način on koriste kao pravila trgovanja te što su to pravila trgovanja općenito. Sljedeće poglavlje 3 opisuje što je to genetsko programiranje i njegove operacije Praktični dio prikazan je i objašnjen u poglavlju 4, dok se objašnjenje o alatima potrebnim za dohvaćanje podataka nalaze dodatku A. Metodologija i rezultati istraživanja prikazani su u poglavlju 5, te zaključak u poglavlju .

[1] eng. initial public offering, IPO

2. Tržišta kapitala

Tržište[1] kapitala je tržište na kojem se trguje financijskim instrumentima. Najčešći oblici kojima se trguje na tržištu kapitala su dionice, obveznice i opcije. Većina ulagača na tržištu kapitala želi ostvariti profit. Profit se ostvaruje ulaganjem u one financijske instrumente za koje se pretpostavlja da će im cijena narasti, te će tako ulagač zaraditi razliku između cijene po kojoj je financijski instrument kupio i one po kojoj ju je prodao.

Za tržišta su karakteristična dva tipa: tržište bikova i tržište medvjeda. Tržište bikova[2] je tržište na kojem cijene dionica rastu ili se očekuje njihov rast. Investitori su optimistični te takvom tržištu kupuju dionice u očekivanju rasta cijena dionica i nastavka dobrih rezultata trgovanja. Tržište medvjeda[3] je suprotno od tržišta bikova te je to tržište na kojem cijene dionica padaju ili se očekuje njihov pad. Investitori na takvom tržištu su pesimistični. Dobar investitor iskorištava stanje i na jednom i na drugom tržištu, npr. kupujući jeftino kade je tržište na dnu i prodaje skupo kada je ono na vrhuncu. Kako bi investitor znao kada je najbolji trenutak za trgovanje on pokušava predvidjeti kretanje tržišta i cijena na njemu.

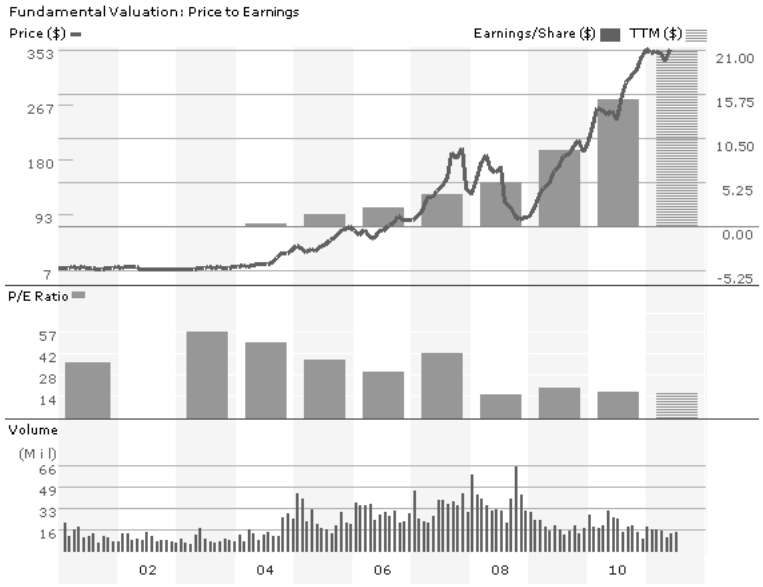
2.1. Predviđanje cijena

Uspješno predviđanje cijena trgovcu donosi značajan profit. No neki vjeruju da je tržište vođeno nasumičnom šetnjom[4] te je stoga nepredvidljivo. Ostali pokušavaju predvidjeti kretanje cijena na tržištu koristeći raznorazne metode. Metode predviđanja grubo su podijeljene u tri kategorije: fundamentalnu analizu (2.1.1), tehničku analizu (2.1.2) i tehnološke metode (2.1.3).

[1] eng. market
[2] eng. bull market
[3] eng. bear market
[4] eng. random walk

2.1.1. Fundamentalna analiza

Fundamentalna analiza[5] proučava utjecaj ekonomskog, političkog i društvenog okruženja na financijsko tržište. Za cilj ima utvrditi stvarnu cijenu tvrtke u koju ulažu, a to postižu analizom raznih ekonomskih parametara kao što su rast, isplata dividendi, visinu kamata, rizik ulaganja, nezaposlenost, inflaciju, štednju, poreze itd. Iz te analize stvarne cijene i trenutne cijene na tržištu dolaze do jednostavnog pravila kada je pogodno vrijeme za uložiti u tvrtku, a kad to nije. Kako do promjena stvarne vrijednosti ne dolazi iz dana u dan fundamentalna analiza nije pogodna za kratkoročnu procjenu kretanja cijena na tržištu. Fundamentalna analiza kaže da je 90% u logičkim parametrima a da je 10% u psihološkim.



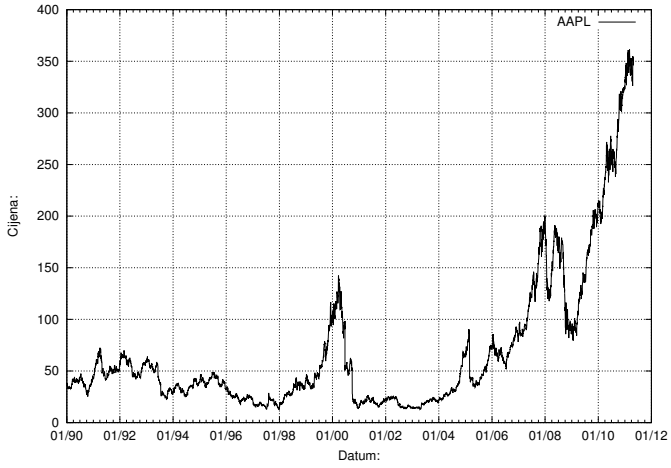
Slika 2.1: Fundamentalna analiza

2.1.2. Tehnička analiza

Tehnička analiza[6] proučava povijesni utjecaj cijena i obima trgovanja na financijsko tržište. Uglavnom se koristi statističkim metodama praćenja trendova, najčešće u obliku grafova. Analitičare tehničke analize ne zanima zašto dolazi do promjene cijene, oni samo reagiraju na nju. Isto tako sama cijena nije glavni interes već njeno kretanje, da li pada ili raste. Tehnička analiza kaže da je 10% u logičkim parametrima a 90% u psihološkim.

[5] fundamental analysis

[6] eng. technical analysis



Slika 2.2: Tehnička analiza

2.1.3. Tehnološke metode

Pojavom digitalnih računala predviđanje cijene na tržištima zakoračilo je na tehnološko područje. Najkorištenije metode pomoću koje za predviđanje cijena su umjetne neuronske mreže[7] i genetski algoritmi[8]. U zadnje vrijeme popularno je i dubinska analiza podataka[9]. Genetsko programiranje kao jedna od tehnoloških metoda obrađena je u ovom radu.

2.1.4. Efikasnost tržišta

Da li je moguće predvidjeti cijene na tržištu? Odgovor na to pitanje pokušali su dati mnogi istraživači. U ekonomiji postoji hipoteza, koju je definirao E.F. Fama 1965, koja kaže da je tržište učinkovito, što podrazumijeva da je nemoguće zaraditi više od ostalih predviđajući cijene na tržištu. Hipoteza također govori da su sve informacije vezane uz tržište sadržane u cijenama i da se sve nove informacije vezane za tržište odražavaju na promjenu cijene na tržištu. Postoje tri oblika hipoteze efikasnosti tržišta:

1. Slab oblik, kaže da su sve informacije o prošlom (povijesnom) kretanju cijena na tržištu sadržane u sadašnjoj cijeni, tj. da ne možemo predvidjeti cijene u budućnosti na temelju cijena u prošlosti. Ovaj oblik hipoteze tvrdi da tehnička analiza (2.1.2) nije u mogućnosti pouzdano predvidjeti kretanje cijena u budućnosti, tj. ostvariti dodatni profit, iako ostavlja mogućnost da pojedini oblici fundamentalne analize to naprave (2.1.1).

[7] eng. artifical neural network, ANN
[8] eng. genetic algorithms
[9] eng. data mining

2. Polujak oblik, kaže da se cijene brzo i adekvatno prilagođavaju svim javno objavljenim informacijama vezanim za poslovanje, tj. da ne možemo koristiti objavljenje podatke za predviđanje cijena u budućnosti. Ovaj oblik hipoteze tvrdi da niti tehnička analiza (2.1.2) niti fundamentalna analiza (2.1.1) nije u mogućnosti pouzdano predvidjeti kretanje cijena u budućnosti, tj. ostvari dodatni profit.

3. Jak oblik, kaže da su sve informacije, i javne i tajne, sadržane u cijeni, tj. da bez obzira koje informacije imali ne možemo predvidjeti cijene u budućnosti.

Hipotezu o efikasnosti tržišta osporavaju zagovornici i fundamentalne analize (2.1.1) i tehničke analize (2.1.2). Vršena su istraživanja na podacima tržišta da bi se dokazalo da su tržišta predvidiva, no nisu pronađeni čvrsti dokazi ni za ni protiv te pretpostavke.

2.2. Pravila trgovanja

Pravilo trgovanja[10] je algoritam koji nam govori kada kupiti (signal 'kupi'[11]), a kada prodati (signal 'prodaj'[12]) određenu dionicu. Kupovanjem dionice nalazimo se na tržištu te dok god dobivamo signal 'kupi' na njemu i ostajemo. Kada dobijemo signal 'prodaj' dionice prodajemo te se tada nalazimo van tržišta i tamo ostajemo dok god dobivamo signal 'prodaj'.

Tehnička analiza (2.1.2) koristi niz jednostavnih pravila za generiranje 'kupi' i 'prodaj' signala, te time i pravila trgovanja. Najčešće korišteni indikatori tehničke analize su: jednostavni i eksponencijalni pomični prosjek (2.2.1), pomični prosjek konvergencije divergencije (2.2.2), stopa promjene (2.2.3), relativni indeks snage (2.2.4) i cjenovni oscilator (2.2.5).

2.2.1. Pomični prosjek

Pomični prosjek[13] je indikator praćenja trenda koji koristi prosjeke zaključnih cijena kako bih olakšao uočavanje trendova. Pomični prosjek se izračunava kao prosjek zaključnih cijena u određenom vremenskom period, tako izračunati prosjek naziva se jednostavni pomični prosjek[14]. Pored jednostavnog pomičnog prosjeka računa se eksponencijalni pomični prosjek[15] koji posljednjim cijenama daje veću težinu a onima

[10] trading rule
[11] eng. buy
[12] eng. sell
[13] eng. moving average, MA
[14] eng. simple moving average, SMA
[15] eng. exponential moving average

koje su prethodile manju težinu. Sam izračun eksponencijalnog pomičnog prosjeka je relativno složen. Svrha pomičnih prosjeka je identificirati signale završetka jednog i/ili početka drugog trenda. Pomični prosjek nikada se ne koristi sam već u kombinaciji sa drugim indikatorima.

2.2.2. Pomični prosjek konvergencije divergencije

Pomični prosjek konvergencije divergencije[16] računa se kao razlika kraćeg i dužeg pomičnog prosjeka. Periodi su proizvoljni, no u praksi se najčešće za duži period uzima 26 dana, a za kraći 12 dana:

$$\text{MACD} = \text{EMA}_{12\text{ dana}} - \text{EMA}_{26\text{ dana}}$$

Koristi se za prepoznavanje stanja ubrzanog rasta/pada cijene ili trenutačne prenapuhanosti potražnje/ponude[17] kao i za predviđanje smjera kretanja cijene. Ukoliko je MACD pozitivan i raste tada takvo kretanje sugerira postojanje uzlaznog trenda, u suprotnom kada je MACD negativan i opada takvo kretanje sugerira postojanje silaznog trenda. MACD se koristi zajedno s 9 dnevnim pomičnim prosjekom, koji se koristi kao linija okidanja. Presijecanje linije okidanja prema gore označava 'kupi' signal, a presijecanje prema dolje 'prodaj' signal. Također se koristi i za uočavanje divergencije naspram zaključne cijene. Kada cijena pada, a MACD počne rasti označava 'kupi' signal, te kada MACD počne padati, a cijena još raste označava 'prodaj' signal.

2.2.3. Stopa promjene

Stopa promjene[18] izračunava se tako da se razlika između posljednje zaključne cijene i zaključne cijene prije n dana podijeli sa zaključnom cijenom prije n dana:

$$\text{ROC} = \frac{\text{C}_{\text{danas}} - \text{C}_{\text{prije n dana}}}{\text{C}_{\text{prije n dana}}}$$

Periodi su proizvoljni, no najčešće se koristi razdoblje od 10 dana. Vrijednosti stope promjene nalaze se oko nulte linije i to ukoliko su pozitivni iznad, a ukoliko su negativni ispod nulte linije. Presijecanje nulte linije prema gore označava 'kupi' signal, a presijecanje prema dolje 'prodaj' signal.

[16] eng. moving average convergence divergence, MACD
[17] eng. overbought/oversold
[18] eng. rate of change, ROC

2.2.4. Relativni indeks snage

Relativni pokazatelj snage[19] uspoređuje snagu porasta cijene dionice i snagu njenih padova u određenom razdoblju. Izračun RSI je relativno složen, no pojednostavljeno koristi se odnos između prosječnih dnevnih porasta cijene i prosječnih dnevnih padova cijene. U praksi se najčešće koristi razdoblje od 14 dana. Koristi za prepoznavanje trenutačne prenapuhanosti potražnje/ponude. Vrijednost RSI-a je između 0 i 100. Umjesto nulte linije ili neke druge linije okidanja za RSI se koristi empirijski utvrđen raspon od 30 i 70. RSI manji od 30 označava 'kupi' signal, a RSI veći od 70 označava 'prodaj' signal.

2.2.5. Cjenovni oscilator

Oscilator volumena[20] mjeri postotnu promjenu cijene između dva pomična prosjeka. Računa se slično kao i MACD, samo što je razlika pomičnih prosjeka podijeljena dužim pomičnim prosjekom i pomnožena sa 100:

$$\text{PPO} = \frac{\text{EMA}_{12\text{ dana}}(V) - \text{EMA}_{26\text{ dana}}(V)}{\text{EMA}_{26\text{ dana}}(V)} \times 100$$

Slično kao i kod ROC indikatora stope variraju oko nulte linije, te presijecanje nulte linije prema gore označava 'kupi' signal, a presijecanje prema dolje 'prodaj' signal.

[19] eng. relative strength index, RSI

[20] eng. percentage price oscillator, PPO

3. Genetsko Programiranje

Genetsko programiranje[1] je stohastička optimiziacijska metoda iz skupine evolucijskih algoritama[2]. Evolucijski algoritmi inspirirani su prirodnim svijetom, poglavito principima iznesenim u Darwinovoj knjizi(Darwin, 1859). Genetsko programiranje prvi uvodi John R. Koza 1999. godine u svojoj knjizi (Koza, 1992). Evolucijski algoritam:

1. Nasumično generiraj početnu populaciju
2. Izračunaj dobrotu jedinke
3. Kreiraj nove jedinke
   (a) Odaberi jedinku/jedinke iz populacije
   (b) Primjeni genetske operacije (križanje, mutacija)
4. Izračunaj dobrotu novih jedinki
5. Ukoliko nije dosegnut uvjet zaustavljanja idi na korak 2

3.1. Prikaz jedinke

Jedinke se mogu prikazati kao programi u nekom funkcijskom jeziku, npr. LISP-u, kao što je to učinjeno u (Koza, 1992), koristeći objektno orijentiranu paradigmu itd. Bez obzira koju metodu koristili jedinke genetskog programa se mogu prikazati kao stabla.

Stablasti prikaz (3.1.1) je najčešće korišteni oblik prikaza genetskog programa, no osim njega koriste se još i linearni prikaz (3.1.2) te prikaz temeljen na grafovima (3.1.3).

[1] eng. genetic programming
[2] eng. evolutionary algorithms

3.1.1. Stablasti prikaz

Stablasti[3] prikaz genetskog programa prikazuje isti u obliku stabla. Stabla su strukture podataka koje se sastoje od čvorova. Svi čvorovi osim prvog, kojeg nazivamo korijenski čvor, imaju čvorove roditelje. Oni čvorovi koji nemaju čvorove djecu nazivaju se čvorovi listovi. Čvorove koji imaju jedno ili više djece zovemo nezavršnim (funkcijskim) čvorovima, a čvorove listove završnim (podatkovnim) čvorovima. U računalnoj znanosti uobičajeno je programe prikazivati pomoću apstraktnog sintaksnog stabla. Kao primjer pokazati ćemo prikaz matematičkog izraza $f(x) = 3 + 2x$ u obliku apstraktnog sintaksnog stabla (slika 3.1), čiji se završni i nezavršni čvorovi nalaze u tablici 3.1.

3.1.2. Linearni prikaz

Linearni[4] prikaz genetskog programa je niz instrukcija imperativnog ili strojnog jezika. U linearno prikazu instrukcije čitaju svoje ulaze iz registara ili memorijskih lokacija te rezultat spremaju u neki od registara. Naspram stablastog prikaza nema razlike između funkcijskih i podatkovnih čvorova jer sve instrukcije obavljaju jednake uloge i komuniciraju putem registara/memorije.

3.1.3. Prikaz grafom

Prikaz temeljen na grafovima[5] sličan je stablastom prikazu, jer su stabla specijalni oblik grafova. Čvorovi su povezani linijama koje određuju i tijek programa i tijek podataka. Na taj način se pojedini dijelovi grafa mogu iskoristiti više puta.

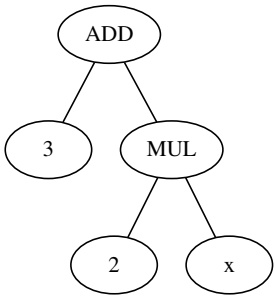
Tablica 3.1: Čvorovi stabla

Nezavršni čvorovi:	ADD, MUL
Završni čvorovi:	2, 3, x

3.2. Početna populacija

Prvi korak evolucijskog algoritma je kreiranje inicijalne populacije. Za genetsko programiranje sa stablastim prikazom to znači izgradnju stabala. Glavni parametar kod

[3] eng. tree based
[4] eng. linear
[5] eng. graph based



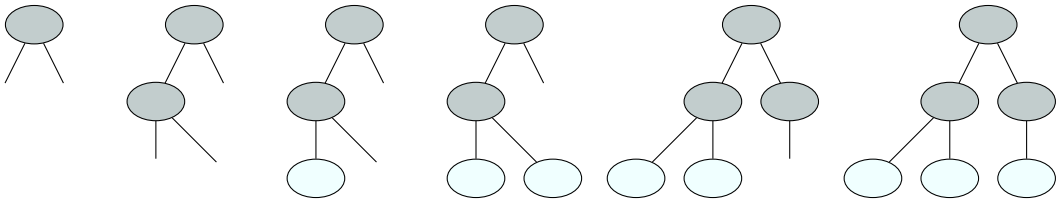
Slika 3.1: Stablasti prikaz

kreiranja stabala je maksimalna dubina. Stabla se izgrađuju nasumičnim odabirom završnih i nezavršnih čvorova. Postupak je rekurzivan a započinje odabirom korijenskog čvora, te dalje nasumičnim odabirom čvorova djece ukoliko je njihov roditelj nezavršni čvor. Postupak se završava kada su svi čvorovi listovi iz skupa završnih čvorova.

Glavne metode izgradnje stabala su puna metoda izgradnje (3.2.1) i rastuća metoda izgradnje (3.2.2). No najkorištenija je pola pola metoda izgradnje (3.2.3) koja kombinira dvije prethodne metode.

3.2.1. Potpuna metoda

Potpuna[6] metoda izgradnje gradi stabla koja su potpuna, tj. dubina svakog čvora koji je list jednaka je odabranoj maksimalnoj dubini. Metoda počinje odabirom korijenskoga čvora iz skupa nezavršnih čvorova. Zatim dalje rekurzivno određuje čvorove djecu na način da ukoliko se čvor nalazi na dubini manjoj od odabrane maksimalne dubine čvor odabire iz skupa nezavršnih čvorova, a ukoliko se čvor nalazi na odabranoj maksimalnoj dubini čvor odabire samo iz skupa završnih čvorova. Stabla koja ova metoda izgrađuje su uglavnom bogatija genetskim materijalom, ali i računski su zahtjevnija.

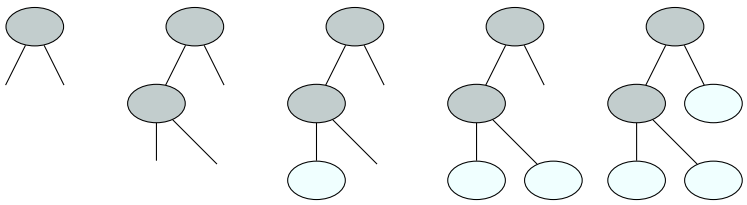


Slika 3.2: Potpuna metoda izgradnje

[6]eng. full

3.2.2. Rastuća metoda

Rastuća[7] metoda izgradnje gradi stabla koja variraju u veličini, od stabala samo sa jednim čvorom do stabala čija je dubina jednaka odabranoj maksimalnoj dubini. Metoda počinje odabirom korijenskoga čvora iz skupa završnih i nezavršnih čvorova. Zatim dalje rekurzivno određuje čvorove djecu na način da ukoliko se čvor nalazi na dubini manjoj od odabrane maksimalne dubine čvor odabire iz skupa završnih i nezavršnih čvorova, a ukoliko se čvor nalazi na odabranoj maksimalnoj dubini čvor odabire samo iz skupa završnih čvorova. Stabla koja ova metoda izgrađuje imaju raznoliku strukturu i manje su računski zahtjevna.



Slika 3.3: Rastuća metoda izgradnje

3.2.3. Pola-pola metoda

Pola-pola[8] metoda gradi stabla na način tako da prvu polovinu populacije gradi potpunom metodom izgradnje, a drugu polovinu populacije gradi rastućom metodom izgradnje. Prilikom izgradnje metoda varira dubinu izgrađenih stabala od najmanje do odabrane maksimalne dubine. Stabla koja ova metoda izgrađuje variraju u veličini i obliku te je ona najčešće korištena metoda izgradnje.

3.3. Mutacija

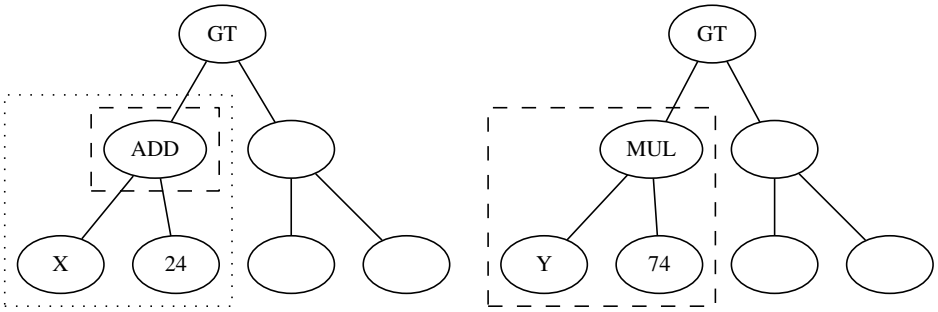
Mutacija[9] je operacija koje se obavlja na pojedinoj jedinki, a unesene promjene predstavljaju novi genetski materijal. Kako se unosi novi materijal operacija mutacije čuva ili čak povećava genetsku raznolikost populacije. No to međutim može dovesti do pada dobrote pojedine jedinke jer mijenja postojeći isprobani genetski materijal.

[7] eng. grow
[8] eng. ramped half-and-half
[9] eng. mutation

U ovom radu razlikovati ćemo četiri različite mutacije: standardnu mutaciju (3.3.1), mutaciju smanjivanjem (3.3.2), mutaciju zamjenom (3.3.3) i mutaciju zamjenom podstabla (3.3.4).

3.3.1. Standardna mutacija

Standardna mutacija[10] unutar jedinke nasumično odabere mjesto mutacije (čvor). Odabranim čvorom definirano je podstablo kojem je odabrani čvor korijenski čvor. Tada se podstablo zamjenjuje novim nasumično generiranim podstablom. Standardna mutacija unosi novi genetski materijal u populaciju.



Slika 3.4: Standardna mutacija

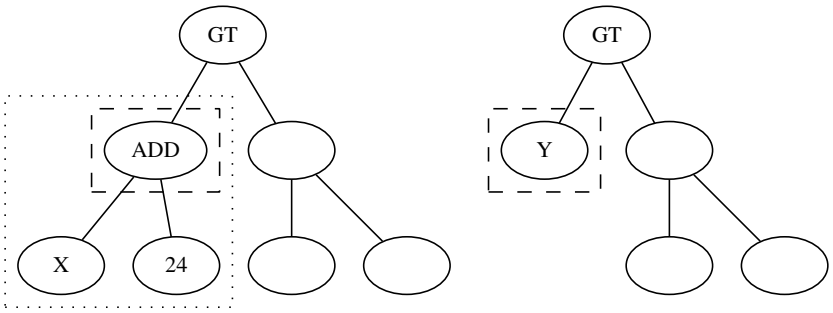
3.3.2. Mutacija smanjivanjem

Mutacija smanjivanjem[11] unutar jedinke nasumično odabere mjesto mutacije (čvor). Odabranim čvorom definirano je podstablo kojem je odabrani čvor korijenski čvor. Tada se podstablo zamjenjuje jednim od čvorova djece tog podstabla. Mutacija smanjivanjem efikasna je metoda smanjivanja kompleksnosti jedinki.

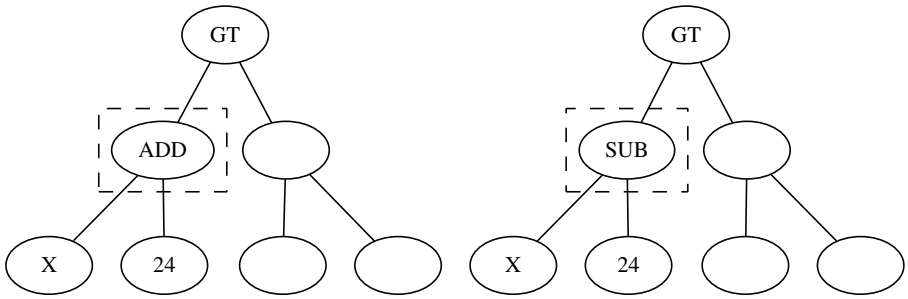
3.3.3. Mutacija zamjenom čvora

Mutacija zamjenom čvora[12] unutar jedinke nasumično odabere mjesto mutacije (čvor). Odabrani čvor se zamjenjuje tako da se funkcija koja odgovara tom čvoru zamjeni funkcijom koja ima isti potpis (broj i tip argumenata, povratni tip, itd.)

[10] eng. standard mutation
[11] eng. shrink mutation
[12] eng. swap mutation



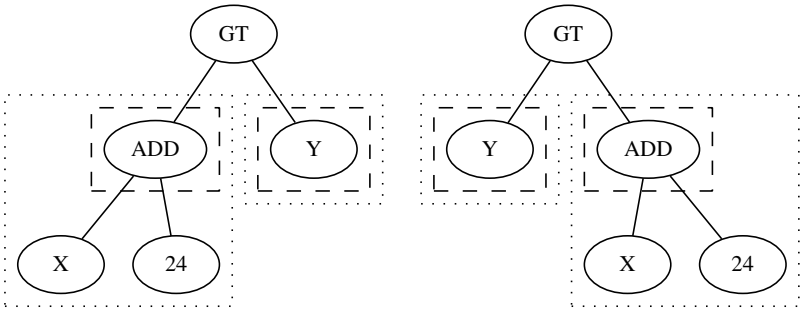
Slika 3.5: Mutacije smanjivanjem



Slika 3.6: Mutacija zamjenom čvora

3.3.4. Mutacija zamjenom podstabla

Mutacija zamjenom podstabla[13] unutar jedinke odabire dva mjesta mutacije (čvora). Odabrani čvorovi definiraju dva podstabla kojima su odabrani čvorovi korijenski čvorovi. Tada se prvo podstablo zamijeni drugim podstablom i drugo podstablo prvim podstablom.



Slika 3.7: Mutacija zamjenom podstabla

[13]eng. swap subtree mutation

3.4. Križanje

Operacija križanja[14] je operacija koja spaja genetski materijal dviju jedinki u novu jedinku. Jedinke koje su pogodne za križanje odabiru se na temelju njihove dobrote. Križanje podstabla dvije jedinke (prvi roditelj i drugi roditelj) obavlja se na način da se u svakoj od jedinki nasumično odabere mjesto križanja (čvor). Odabrani čvorovi definiraju dva podstabla, po jedno podstablo u svakom roditelju, kojima su odabrani čvorovi korijenski čvorovi. Tada se kreira nova jedinka (dijete) koja nastaje tako da se podstablo iz prvog roditelja zamijeni podstablom iz drugog roditelja. Dijete ima genetski materijal i prvog i drugog roditelja.

3.5. Odabir

Odabir[15] je operacija kojom se iz populacije odabire jedna ili više jedinka na kojima se obavljaju operacije križanja i mutacije. Broj jedinki koji se tom prilikom zamjenjuje naziva se selekcijski pritisak. Ukoliko je selekcijski pritisak mali, brzina konvergencije je također mala te će evolucijskom algoritmu biti potrebno više vremena za pronalazak rješenja. No ako je selekcijski pritisak prevelik postoji mogućnost da će evolucijski algoritam prerano konvergirati prema rješenju, često suboptimalnom. Najčešće korištena selekcija je N-turnirski odabir (3.5.2) no koriste se još jednostavni odabir (3.5.1), nasumični odabir (3.5.3) i leksička škrtost (3.5.4).

3.5.1. Jednostavni odabir

Jednostavni odabir [16] svakoj jedinki dodjeljuje se vjerojatnost odabira koja je proporcionalna njenoj dobroti:

$$p_(i) = \frac{d_i}{\sum_{j=1}^{N} d_j}$$

gdje je d - dobrota pojedine jedinke, a N - broj jedinki u populaciji. To možemo predstaviti kotačom roulette-a gdje je svakoj jedinki dodijeljen dio kotača obrnuto proporcionalan vjerojatnosti odabira. Kako je najboljoj jedinki dodijeljen najmanji dio kotača vjerojatnost da bude odabrana je mala, nasuprot tome najlošijoj jedinci dodijeljen je najveći dio kotača te je vjerojatnost da bude odabrana velika.

[14] eng. crossover
[15] eng. selection
[16] eng. roulette

3.5.2. K-turnirski odabir

K-Turnirski [17] odabir iz populacije nasumično odabere nekoliko jedinki (K - veličina turnira), obično je veličina turnira mala naspram veličine populacije. Jedinke se međusobno uspoređuju na temelju dobrote, te se odabire najbolja. Turnirski odabir kod kojeg je veličina turnira 1 ekvivalentan je nasumičnom odabiru jedinke iz populacije, a kada je veličina turnira jednaka veličini populacije odabire se najbolja jedinka u populaciji.

3.5.3. Nasumični odabir

Nasumični[18] odabir kako mu i samo ime kaže iz populacije se nasumično odabere jedinku.

3.5.4. Leksička škrtost

Leksička škrtost[19] se vodi principom Occamove oštrice (britve) koja kaže 'ako dvije hipoteze podjednako dobro mogu objasniti neku pojavu, mora se izabrati ona koja zahtijeva manje pretpostavki'. Što pojednostavnjeno znači da ukoliko postoje dva rješenja, treba odabrati ono jednostavnije. Postupak odabira identičan je K-turnirskom odabiru (3.5.2) samo što se u slučaju da dvije jedinke imaju jednaku dobrotu odabiremo onu koja je manja (jednostavnija).

3.6. Dobrota

Kako bi evolucijski algoritam mogao odrediti koje je najbolje rješenje potrebno je uvesti mjeru jedinke koju nazivamo dobrotom jedinke. Najčešće je dobrota jedinke definirana kao numerička vrijednost koja predstavlja udaljenost od željenog rješenja. Kako je željeno rješenje specifično za svaki problem koji se rješava, ne može se dati generalno pravilo kako definirati dobrotu.

Dobrotu možemo podijeliti na nekoliko tipova: sirovu (3.6.1), standardiziranu (3.6.2), prilagođenu (3.6.3) i normaliziranu dobrotu (3.6.4).

[17] eng. tournament
[18] eng. random
[19] eng. parsimony tournament

3.6.1. Sirova dobrota

Sirova[20] dobrota definirana je prirodom problema. Tako u jednom problemu veće vrijednosti dobrote mogu karakterizirati bolje jedinke, dok u drugom problemu manje vrijednosti mogu karakterizirati bolje jedinke. Sirovom dobrotom ne možemo jednoznačno okarakterizirati bolje i loše jedinke.

3.6.2. Standardizirana dobrota

Standardizirana[21] dobrota slična je sirovoj dobroti s ograničenjem da vrijednosti ne smiju biti manje od nule, a manja vrijednost karakterizira bolju jedinku. Često se najbolje jedinke karakteriziraju s vrijednošću 0. Koristeći standardiziranu dobrotu možemo utvrditi koliko je pojedina jedinka bolja od druge, no ne možemo utvrditi koliko. Standardiziranu dobrotu možemo izračunati iz sirove dobrote. Ukoliko kod sirove dobrote veća vrijednost karakterizira bolju jedinku, standardiziranu dobrota dobiva se direktno iz sirove dobrote, te ukoliko postoje vrijednosti manje od nule pomaknemo sve vrijednosti za konstantu P. Ukoliko manja vrijednost karakterizira bolju jedinku standardiziranu dobrotu računamo iz sirove dobrote uz promjenu predznaka i pomak za konstantu P ukoliko je potreban.

3.6.3. Prilagođena dobrota

Prilagođena[22] dobrota se računa iz standardizirane dobrote pomoću izraza: $d_p = \frac{1}{1+d_{s_k}}$. Vrijednosti prilagođene dobrote su na intervalu $[0, 1]$, a veća vrijednost karakterizira bolju jedinku. Prilagođena dobrota dobro prikazuje male razlike između dobrota jedinki te tijekom generacija ta razlika biva sve više izražena.

3.6.4. Normalizirana dobrota

Normalizirana[23] dobrota se računa iz prilagođene dobrote pomoću izraza: $d_n = \frac{d_p}{\sum_k d_{p_k}}$. Vrijednosti normalizirane dobrote su na intervalu $[0, 1]$, veće vrijednost karakterizira bolju jedinku. Suma normaliziranih dobrota je 1.

[20] eng. raw fitness
[21] eng. standardized fitness
[22] eng. adjusted fitness
[23] eng. normalized fitness

3.7. Zaustavljanje

Kriteriji zaustavlanja[24] evolucijskog algoritma bi u idealnom slučaju bio kada smo dosegli željeno rješenje. No međutim kako se u većini slučajeva asimptotski približavano idealnom rješenju te ga ne dosežemo u prihvatljivom vremenu kriterij zaustavljanja prilagođavamo realnim uvjetima. Tako npr. uvjet može biti zaustavljanje nakon određenog broja generacija ili dosezanja određene vrijednosti dobrote. Jedna od metoda pravovremenog zaustavljanja je i ispitivanje stagnacije. Ukoliko tijekom zadnjih n generacija nije došlo do poboljšanja populacije prekida se algoritam evolucije.

3.8. Strogo tipizirano genetsko programiranje

Strogo tipizirano genetsko programiranje[25] je nadopuna (Montana, 1995) genetskog programiranja na način da sve povratne vrijednosti funkcija i argumenti ne moraju biti istoga tipa kako to specificira (Koza, 1992). Miješanje različitih tipova podataka olakšava nam pristup problemu jer možemo baratati prirodnijim funkcijama. Jedan od primjera je nezavršni čvor IF-THEN-ELSE, na slici 3.9, koji kao uvjet ispituje čvor koji mora biti logičkog (bool) tipa, na temelju kojeg vraća cjelobrojnu vrijednost (integer) i to ukoliko je logički čvor istinit vraća drugi čvor, a ukoliko je neistinit vraća treći čvor. Kod strogo tipiziranog gentskog programiranja svaki završni čvor ima definiran tip, a svaki nezavršni čvor ima definiran tip za svaki argument i tip povratne vrijednosti. Strogo tipizirano genetsko programiranje uvodi dodatna dva kriterija kojima se određuje što definira ispravno sintaksno stablo genetskog programa.

1. Korijenski čvor stabla vraća vrijednost tipa koja je definirana rješenjem problema.

2. Svaki ne korijenski čvor vraća vrijednost onog tipa koju zahtjeva njegov roditeljski čvor.

Kako bi stabla koja predstavljaju jedinke populacije bila sintaksno ispravna potrebno je da metode izgradnje (3.2), operacije mutacije (3.3) i operacije križanja (3.4) poštuju navedene kriterije.

[24] eng. termination criteria

[25] strongly typed genetic programming

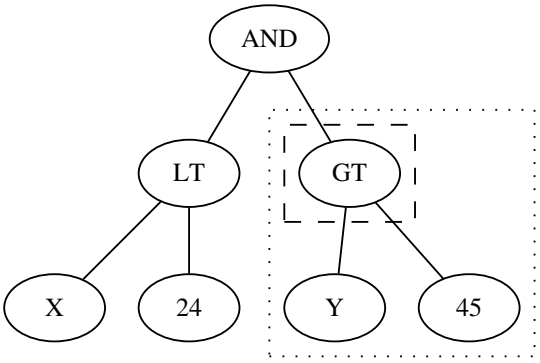
3.9. Genetsko programiranje i strojno učenje

Strojno učenje[26] je dio računarske znanosti koji se bavi razvojom algoritama koji omogućuju računalima da unaprijede i poboljšaju vlastito ponašanje. Jedna od korištenih tehnika je učenje stablima odluke[27]. U toj tehnici se stablastom strukturom zapisuju zapažanja o podacima za učenje kako bi se kasnije mogla donijeti odluka o ispitnim podacima. Kako su opažanja opisana stablastim strukturama genetsko programiranje je pogodno za rješavanje takvih problema. Uz stabla odluke vezujemo problem prenaučenosti podataka. Prenaučenost je pojava kada se algoritam previše prilagodi na skup podataka na kojem uči te pokazuje slabije rezultate na neviđenim skupovima podataka. Kako bi se riješio taj problem skup za učenje se dijeli na:

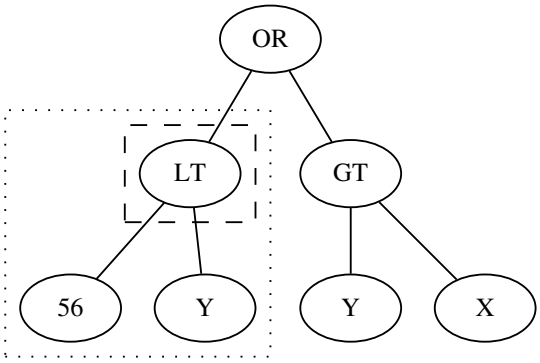
1. skup za učenje[28]
2. skup za provjeru[29]

Skup za provjeru koristi se za prekid učenja ukoliko je došlo do prenaučenosti.

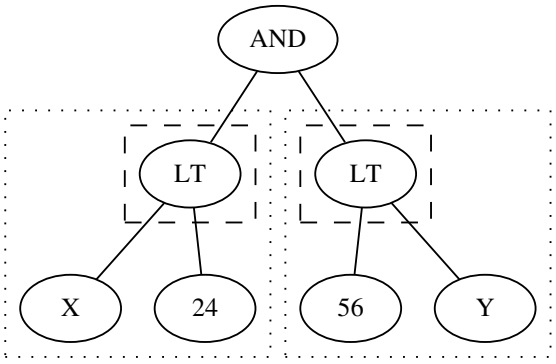
[26] eng. machine learning, ML
[27] eng. decision trees
[28] eng. training set
[29] eng. validation set



(a) Prvi roditelj

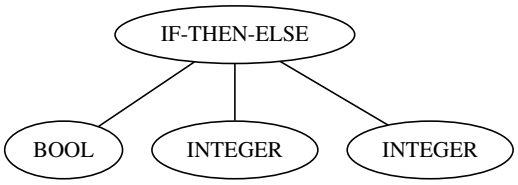


(b) Drugi roditelj



(c) Dijete

Slika 3.8: Križanja dviju jedinki



Slika 3.9: Čvor koji barata sa različitim tipovima podataka

4. Programsko ostvarenje

Programska ostvarenje genetskog programiranja ostvarena je u programskom jeziku C++ korištenjem OpenBeagle biblioteke funkcija za implementaciju genetskog programiranja.

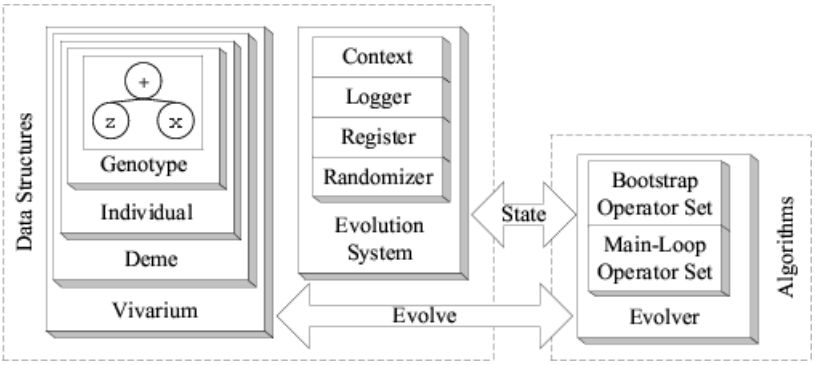
4.1. Openbeagle

Open Beagle[1] je biblioteka funkcija napisana u C++ programskom jeziku. Arhitekturalno se oslanja na objektno orijentiranu paradigmu te tako svojim korisnicima omogućuje vrlo jednostavnu nadogradnju ugrađenih funkcionalnosti. Genetske operacije implementirane su putem C++ koda, a sam evolucijski algoritam implementira se putem XML datoteke. Arhitektura Open Beaglea prikazana je na slici 4.1. Nove genetske operacije, tipovi prikaza jedinki itd. implementiraju se u C++ programskom jeziku nadgradnjom postojećih baznih klasa. Implementacija evolucijskog algoritma putem XML datoteke omogućuje korisniku veću fleksibilnost prilikom testiranja jer nije potrebno rekompajlirati izvorni kod kod projekta kako bi se izmijenio evolucijski algoritam, već je dovoljno samo izmijeniti konfiguracijsku datoteku. Konfiguracijska datoteka osim algoritma evolucije sadrži i evolucijske parametre kao što su veličina populacije, vjerojatnosti pojedinih mutacija itd.. Najvažniji parametri evolucije nalaze se u tablici 4.1. Evolucijske parametre osim putem konfiguracijskoj datoteke možemo proslijediti i putem naredbenog retka:

```
prog.exe -OBparametar1=vrijednost1,parametar2=vrijednost2
```

Algoritam evolucije u konfiguracijskoj datoteci sastoji se od dva djela. Prvi dio je inicijalizacijski dio u kojemu se kreira inicijalna populacija ili se učitava stanje neke prethodne evolucije spremljene u datoteci. Drugi dio je glavna petlja evolucijskog algoritma koji se nalazi unutar <MainLoopSet></MainLoopSet> odjeljka XML datoteke. Operacije genetskog programa koje se mogu koristiti za gentsko programiranje opisane su u tablici 4.2. Primjer evolucije s jednostavnim odabirom, križanjem i standardnom mutacijom prikazan je u listingu 4.2.

[1] `http://beagle.gel.ulaval.ca/`



Slika 4.1: Arhitektura Open Beaglea

Tablica 4.1: Evolucijski parametri

Parametar	*Tip*	*Pretpostavljena vrijednost*	*Opis*
ec.conf.file	String	prog.conf	Ime XML datoteke koja sadrži konfiguraciju programa
ec.pop.size	Int	100	Broj jedinki u populaciji, ukoliko se koristi više populacija parametar se navodi kao polje vrijednosti odvojenih s / (npr.: 100/200/200)
ec.term.maxgen	Int	50	Maksimalni broj generacija u evoluciji (jedan od uvjeta zaustavljanja) (3.7)
gp.tree.maxdepth	Int	17	Maksimalna dubina stabla
gp.init.maxdepth	Int	5	Maksimalna dubina novo generiranih stabala
gp.init.mindepth	Int	2	Minimalna dubina novo generiranih stabala
gp.mutstd.indpb	Float	0.05	Vjerojatnost standardne mutacije (3.3.1)
gp.mutshrink.indpb	Float	0.05	Vjerojatnost mutacije smanjivanjem (3.3.2)
gp.mutswap.indpb	Float	0.05	Vjerojatnost mutacije zamjenom (3.3.3)
gp.mutsst.indpb	Float	0.0	Vjerojatnost mutacije zamjenom podstabla (3.3.4)
gp.mutstd.maxdepth	Int	5	Maksimalna dubina čvora na kojem se obavlja standardna mutacija
gp.mutswap.distrpb	Float	0.5	Vjerojatnost da je točka mutacije nezavršni (funkcijski) čvor. Vrijednost 1.0 znači da su sve točke mutacije nezavršni čvorovi, 0.0 znači da su sve točke mutacije završni čvorovi (listovi)
gp.mutsst.distrpb	Float	0.5	Vjerojatnost interne ili externe mutacije kod mutacije zamjenom podstabla Vrijednost 1.0 znači da je mutacija zamjenom stabla interna (druga točka je unutar točke podstabla prve točke), 0.0 znači da je mutacija externa (obje točke su izvan podstabla tih točaka)
gp.cx.indpb	Float	0.05	Vjerojatnost križanja (3.4)

gp.cx.distrpb	Float	0.9	Vjerojatnost križanja da je točka križanja nezavršni (funkcijski) čvor. Vrijednost 1.0 znači da su sve točke križanja nezavršni (funkcijski) čvorovi, 0.0 znači da su sve točke križanja završni čvorovi (listovi).
gp.try	Int	2	Maksimalni broj pokušaja modifikacije stabla genetskog programa
ms.write.interval	Int	0	Interval učestalosti snimanja trenutnog statusa evolucije (broj generacija), ukoliko je 0 znači snimanje na kraju evolucije
ms.restart.file	String		Ime datoteke koja sadrži podatke o trenutnom statusu evolucije iz kojeg se evolucija može nastaviti
ms.write.over	Bool	0	Određuje da li će se datoteke sa trenutnim statusom evolucije prepisivati nakon svakog intervala ili će za svaki interval biti kreirana nova
lg.console.level	Int	2	Količina informacija ispisana na konzolu

Tablica 4.2: Operacije evolucije

Operator	Značenje
Kreiranje inicijalne populacije:	
<GP-InitFullConstrainedOp/>	Potpuna metoda izgradnje (3.2.1)
<GP-InitGrowConstrainedOp/>	Rastuća metoda izgradnje (3.2.2)
<GP-InitHalfConstrainedOp/>	Pola-pola metoda izgradnje (3.2.3)
Križanje:	
<GP-CrossoverConstrainedOp/>	Križanje (3.4)
Mutacija:	
<GP-MutationStandardConstrainedOp/>	Standardna mutacija (3.3.1)
<GP-MutationShrinkConstrainedOp/>	Mutacija smanjivanjem (3.3.2)
<GP-MutationSwapConstrainedOp/>	Mutacija zamjenom čvora (3.3.3)
<GP-MutationSwapSubtreeConstrainedOp/>	Mutacija zamjenom podstabla (3.3.4)
Odabir:	
<SelectRandomOp/>	Nasumični odabir (3.5.3)
<SelectTournamentOp/>	Turnirski odabir (3.5.2)
<SelectRouletteOp/>	Jednostavni odabir (3.5.1)
<SelectParsimonyTournOp/>	Leksikografska škrtost (3.5.4)
Uvjet zaustavljanja:	

<TermMaxGenOp>	Maksimalni broj generacija
<TermMaxFitnessOp>	Maksimalna dobrota
<TermMinFitnessOp>	Minimalna dobrota
<TermMaxEvalsOp>	Maksimalan broj evaluacija
<TermMaxHitsOp>	Maksimalan broj pogodaka
Evaluacija:	
<eval/>	Računa dobrotu jedinke (3.6)

4.2. Implementacija

Kako ne bi došlo do kolizije u imenima postojećih i dodatno implementiranih klasa, dodatno implementirane klase nalaze se u posebnom prostoru imena: trading. Da bi se mogla pratiti dobrota na skupu za učenje kao i na skupu za provjeru implementirana je nova klasa (trading :: Fitness) koja nasljeđuje postojeću klasu za dobrotu (Beagle :: FitnessSimple) te pored dobrote na skupu za učenje pamti i dobrotu na skupu za provjeru, što je pogodno za pregled kretanja dobrote na oba skupa. Statističko praćenje kretanja dobrote na oba skupa implementirano je dodatnom klasom (trading :: StatsCalcOp), a nasljeđuje postojeću klasu (Beagle :: GP :: StatsCalcFitnessSimpleOp) koja uvodi novu operaciju GP-StatsCalcTrading. Operacija dodatno računa statistike (minimalnu i maksimalnu vrijednost, srednju vrijednost i standardnu devijaciju doborte cijele populaciji na skupu za provjeru), dok statistike na skupu za učenje računa bazna klasa. Funkcijski i podatkovni čvorovi (4.3) izgrađeni su nadogradnjom bazne klase za čvorove (Beagle :: GP :: Primitive) te implementiraju zadane funkcije odnosno podatke. Kako bi se pojednostavio kod, veći dio koda koji obavlja računanja indikatora tehničke analize izdvojen je u posebnu datoteku *ta.cpp*. Operacija evaluacije implementirana je u datoteci *eval.cpp*, a putem funkcije eval :: evaluate_interval (GP :: Individual& inIndividual, GP :: Context& ioContext) evaluira se pojedina jedinka na zadanom intervalu. Intervali se postavljaju funkcijom eval :: set_testing_interval (std :: string start, std :: string end) ili skraćeno za skup za učenje eval :: set_training_interval() i eval :: set_validation_interval() za skup za provjeru. Kako bi evaluacijska funkcija znala koju dionicu, na koji dan i iz koje baze podataka evaluira implementirana je klasa za čuvanje konteksta (trading :: Context) izvedena iz bazne klase za kontekst (Beagle :: GP :: Context). Tijekom evolucije najbolje jedinke cijele evolucije čuvaju

se u 'dvorani slavnih'[2]. Nakon dosezanja kriterija zaustavljanja (3.7) iz dvorane slavnih se izabire najbolja jedinka na temelju dobrote na skupu za provjeru te se zapisuje u datoteku *best.xml*. Implementacija čita konfiguracijsku datoteku *prog.conf*.

Nakon generiranja pravila ono se putem programa *ind* može testirati na bilo kojoj drugoj dionici i na bilo kojem vremenskom intervalu. Program za testiranje pravila čita konfiguracijsku datoteku *ind.conf*, no potrebno je samo specificirati odgovarajuće parametre iz tablice 4.4 bez algoritma evolucije jer se jedinka samo evaluira. Nakon izvođenja, program za testiranje ispisuje dobrotu jedinke na testiranom intervalu.

4.3. Čvorovi

Za generiranje pravila trgovanja odabrane su operacije iz tablice 4.3. Zahtjeva se da je korijenski čvor logičkog tipa te se koristi strogo tipizirano genetsko programiranje (3.8).

Tablica 4.3: Čvorovi

Nezavršni (funkcijski)	
Aritmetičke operacije	+ (zbrajanje, ADD), - (oduzimanje, SUB), * (množenje, MUL), / (dijeljenje, DIV)
Funkcije usporedbe	< (manje od, LT), > (veće od, GT), = (jednako, EQ)
Logičke operacije	i (AND), ili (OR), ne (NOT)
Logička funkcija	Ako-onda-inače (IF-THEN-ELSE)
Aritmetičke funkcije	prosječna vrijednost u periodu (AVG), minimalna vrijednost u periodu (MIN), maksimalna vrijednost u periodu (MAX)
Financijski pokazatelji	ROC (2.2.3), RSI (2.2.4), EMA (2.2.1), MACD (2.2.2), PPO (2.2.5)
Završni (podatkovni)	
Logičke konstante	istina (TRUE), laž (FALSE)
Cjelobrojna konstanta	dana (D), kao argument AVG, MIN, MAX [0,260]
Realne konstante	E ([-1,1])
Parametri	cijena (P), količina (V)

4.4. Dobrota jedinke

Dobrota generiranog pravila je definirana kao zarada povrh zarade koju ostvaruje bazna strategija s kojim se generirano pravilo uspoređuje. Programski su implementirana

[2]eng. hall of fame

dvije strategije: 'buy and hold' strategija i 'nasumična' strategija. 'Buy and hold' strategija je definirana kupovinom na početku perioda i prodajom na kraju perioda u kojem se trguje. Zarada koja se ostvaruje je rezultat razlike u cijeni po kojoj se kupovalo i one po kojoj se prodavalo. Nasumična strategija je definirana kupovinom i prodajom nasumično. Kako generirano pravilo trgovanja i nasumična strategija mogu na završetku perioda još uvijek biti na tržištu, definirano je da se na kraju perioda izlazi sa tržišta (prodaju se dionice). Dobrota je definirana na način kako su je i u svom radu definirali Allen i Karjalainen (1999). Zarada jednog trgovanja je:

$$\pi_i = \frac{P_{s_i}}{P_{b_i}} \times \frac{1-c}{1+c} - 1 = \exp\left[\sum_{t=b_i+1}^{s_i} r_t + \log\frac{1-c}{1+c}\right] - 1$$

gdje je P_{s_i} cijena prilikom prodaje, P_{b_i} cijena prilikom kupovine te c provizija (izražena kao dio cijene). A dnevna zarada definirana je kao zarada koja se ostvaruje između dva tržišna dana:

$$r_t = \log P_t - \log P_{t-1}$$

gdje je P_t cijena na dan t i P_{t-1} cijena na dan t-1 (prethodni dan). Iz toga slijedi da je zarada koju ostvaruje 'buy and hold' pravilo jednaka sumi svih dnevnih zarada:

$$r_{bh} = \sum_{t=1}^{T} r_t + \log\frac{1-c}{1+c}$$

umanjena za proviziju c, gdje T označava ukupan broj dana trgovanja. Zarada koju ostvaruje generirano pravila jednaka je sumi dnevnih zarada ali samo za one dane u kojima je pravilo na tržištu:

$$r_r = \sum_{t=1}^{T} r_t I(t) + n\log\frac{1-c}{1+c}$$

gdje je n broj transakcija, a $I(t)$ definiran na sljedeći način:

$$I(t) = \begin{cases} 1 & \text{na tržištu} \\ 0 & \text{van tržišta} \end{cases}$$

Ukupna zarada definirana je kao:

$$\pi = e^r - 1$$

Stoga je zarada povrh 'buy and hold' pravila:

$$\Delta r = r_r - r_{bh}$$

te je dobrota definirana kao:

$$d = e^{\Delta r}$$

4.5. Parametri programa

Osim standardnih parametara evolucije (tablica 4.1) definirani su i parametri specifični za implementaciju (tablica 4.4). Parametrima se definira skup za učenje, skup za provjeru, bazna strategija prema kojoj se računa dobrota jedinke, ime dionice kojom se trguje i ime SQLite baze podataka koja sadrži potrebne podatke o povijesnim podacima dionice. U izradi ovog rada analizirani su podaci sa Zagrebačke Burze[3] i Yahoo! Finance portala[4], no sama implementacija je dovoljno generička da se mogu analizirati podaci s bilo koje burze. Sve što je potrebno je implementirati skripte za dohvaćanje podataka. Više o samoj implementaciji baze podataka i o implementiranim skriptama za učitavanje podataka nalazi se u dodatku A.2. Potrebno je da podaci baze podataka sadrže barem 260 radnih dana podataka prije početnog datuma trgovanja za skup za učenje kako bi funkcije koje koriste vremenske intervale bile ispravno definirane.

Tablica 4.4: Parametri specifični za implementaciju

Parametar	*Tip*	*Pretpostavljena vrijednost*	*Opis*
trading.database	String	se.db	Datoteka sa SQLite bazom podataka iz koje se čitaju podaci o dionicama
trading.ticker	String	AAPL	Ime dionice za koje se generira pravilo trgovanja
trading.ts_date	String	2000-01-01	Početni datum trgovanja za skup za učenje. Datum oblika GGGG-MM-DD[5].
trading.te_date	String	2005-31-12	Završni datum trgovanja za skup za učenje. Datum oblika GGGG-MM-DD.
trading.vs_date	String	2006-01-01	Početni datum trgovanja za skup za provjeru. Datum oblika GGGG-MM-DD.
trading.ve_date	String	now	Završni datum trgovanja za skup za provjeru (validation set). Datum oblika GGGG-MM-DD ili now što označava datum prilikom pokretanja programa.
trading.fee	Double	0.0025	Naknada za svaku transakciju u postotku od cijene.
trading.strategy	Integer	1	Odabir bazne strategije prema kojoj se uspoređuju generirana pravila (1 buy-and-hold, 2 nasumično kupovanje i prodavanje).
trading.calc_vs	Bool	1	Određuje da li se računa dobrota na skupu za provjeru.
trading.best	String	best.xml	Ime datoteke u koju se upisuje/čita najbolja jednika.

[3] `http://www.zse.hr`
[4] `http://finance.yahoo.com`
[5] GGGG godina, MM mjesec, DD dan

Listing 4.1: Primjer XML konfiguracijske datoteke

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Beagle version="3.0.3">
  <Register>
    <Entry key="parametar1">vrjednost1</Entry>
    <Entry key="parametar2">vrjednost2</Entry>
    <Entry key="parametar3">vrjednost3</Entry>
  </Register>
  <Evolver>
    <BootStrapSet>
      <IfThenElseOp parameter="ms.restart.file" value="">
        <PositiveOpSet>
          <Kreiranje_incijalne_populacije/>
          <Evaluacija/>
          <Statistike/>
        </PositiveOpSet>
        <NegativeOpSet>
          <MilestoneReadOp/>
        </NegativeOpSet>
      </IfThenElseOp>
      <TermMaxGenOp/>
      <MilestoneWriteOp/>
    </BootStrapSet>
    <MainLoopSet>
      <Operator_odabira/>
      <Operator_krizanja/>
      <Operator_mutacije/>
      <Evaluacija/>
      <Statistike/>
      <UvjetZaustavljanja/>
      <MilestoneWriteOp/>
    </MainLoopSet>
  </Evolver>
</Beagle>
```

Listing 4.2: Primjer evolucije

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Beagle version="3.0.3">
  <Evolver>
    <MainLoopSet>
      <SelectRouletteOp/>
      <GP-CrossoverConstrainedOp/>
      <GP-MutationStandardConstrainedOp/>
      <eval/>
      <GP-StatsCalcFitnessSimpleOp/>
      <TermMaxGenOp/>
      <MilestoneWriteOp/>
    </MainLoopSet>
  </Evolver>
</Beagle>
```

5. Rezultati

5.1. Utjecaj evolucijskih parametara

Kako bi ocijenili uspješnost traženja pravila trgovanja generiranih putem genetskog programiranja u ovisnosti o evolucijskim parametrima izvršeno je ispitivanje višestrukim pokretanjem programa varirajući pri tome razne evolucijske parametre. Obavljeno je ukupno tri skupa testiranja od po 90 pokretanja za svaki skup. Za svaki skup varirani su svi parametri (kartezijev produkt) iz tablice 5.1. Vremenski interval od 01.01.2000 do 31.12.2010 podijeljen je na dva dijela. Prvi dio koji čini oko 70% iskorišten je za skup za učenje, a preostali dio od 30% za skup za ispitivanje. Ukoliko su jedinke nakon testiranja imale dobrotu 1 ili manju, bilo na skupu za učenje bilo na skupu za ispitivanje, odbačene su iz daljnje analize rezultata jer kao takve nisu našle rješenje bolje od bazne strategije s kojom su uspoređivane. Od ukupno 270 mogućih rezultata nakon izbacivanja nezanimljivih rješenja ostalo je 110 rezultata što je oko 40% ukupnog broja rezultata te su ona uzeta u daljnje razmatranje. Statistički pregled dobivenih rješenja nalazi se u tablici 5.2.

Kretanje vrijednosti dobrota rezultata uzetih u razmatranje na skupu za učenje i na skupu za ispitivanje u odnosu na broj generacija prikazano je na slikama 5.2 i 5.3, u odnosu na veličinu populacije na slikama 5.4 i 5.5, te u odnosu na mutaciju na slikama 5.6 i 5.7.

Najbolja jedinka prikazana je na slici 5.1, a korišteni evolucijski parametri su: 200 jedinki u populaciji, 30 generacija i standardna mutacija od 0.1. Dobrota najbolje jedinke na skupu za ispitivanje je $d_i = 1.70235$, a na skupu za učenje je $d_u = 6.96275$.

5.2. Utjecaj skupa za provjeru

Kako bi utvrdili utjecaj odabira na skupu za provjeru korišteni su evolucijski parametri iz tablice 5.3. Obavljeno je ukupno dva skupa testiranja od po 30 pokretanja za svaki

Tablica 5.1: Parametri testiranja (utjecaj evolucijskih parametara)

Broj generacija	10, 20, 30, 50, 100, 200
Veličina populacije (broj jedinki)	5, 10, 15, 20, 30
Standardna mutacija	0.01, 0.05, 0.1
Vremenski period (skup za učenje)	01.01.2000 do 31.12.2007
Vremenski period (skup za ispitivanje)	01.01.2008 do 01.01.2011
Dionica	AAPL (Apple inc.)
Strategija za usporedbu	'buy and hold'

Tablica 5.2: Statistike rezultata (utjecaj evolucijskih parametara)

	Avg	Stddev	Max	Min
Dobrota na skupu za učenje	3.2	0.99	6.96	1.01
Dobrota na skupu za ispitivanje	1.21	0.17	1.7	1.01
Dubina stabla	4.62	2.14	16.0	3.0
Veličina stabla	12.1	17.91	159.0	4.0

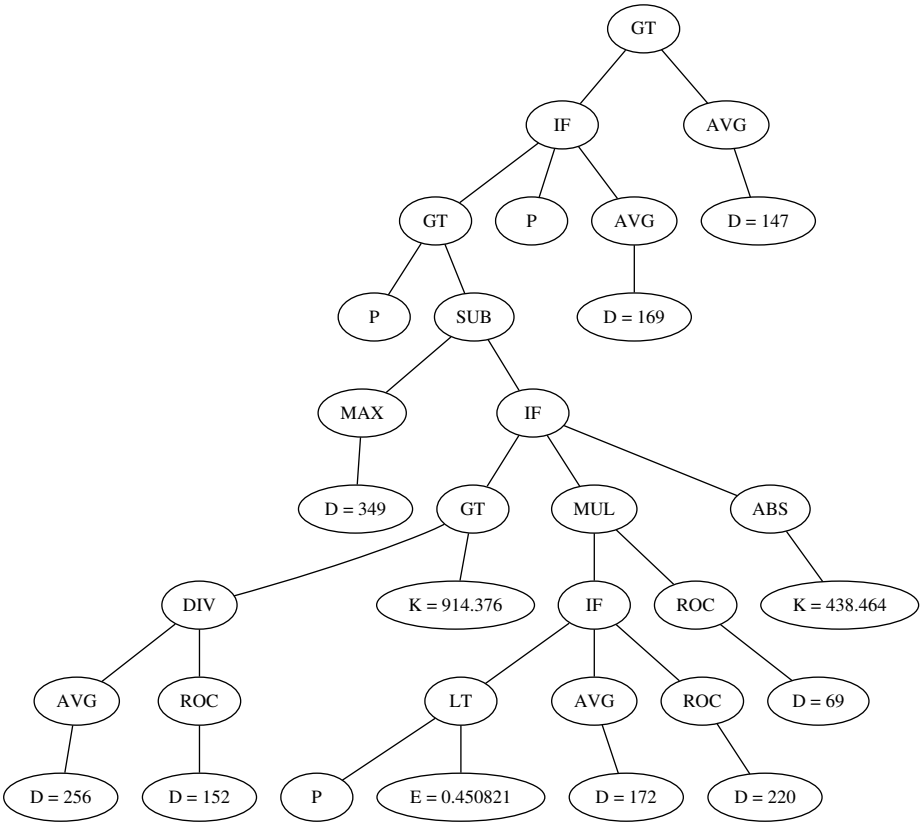
skup. Prva skupina ispitivanja obavljena je na način da je na kraju evolucije kao najbolja jedinka odabrana ona jedinka koja je imala najbolju dobrotu na skupu za učenje (5.2.1). Druga skupina ispitivanja obavljena je na na način da je na kraju evolucije kao najbolja jedinka odabrana ona jedinka koja je imala najbolju dobrotu na skupu za provjeru (5.2.2).

Tablica 5.3: Parametri testiranja (utjecaj skupa za provjeru)

Broj generacija	100
Veličina populacije (broj jedinki)	20
Standardna mutacija	0.1
Dionica	AAPL (Apple inc.)
Strategija za usporedbu	'buy and hold'

5.2.1. Bez skupa za provjeru

Vremenski interval od 01.01.2000 do 31.12.2010 podijeljen je na dva dijela. Prvi dio koji čini oko 70% iskorišten je za skup za učenje, a preostali dio od 30% za skup za ispitivanje. Statistički pregled dobivenih rješenja nalazi se u tablici 5.5. Najbolja jedinka prikazana je na slici 5.8. Dobrota najbolje jedinke na skupu za ispitivanje je $d_i = 1.59373$, a na skupu za učenje je $d_u = 6.0652$.



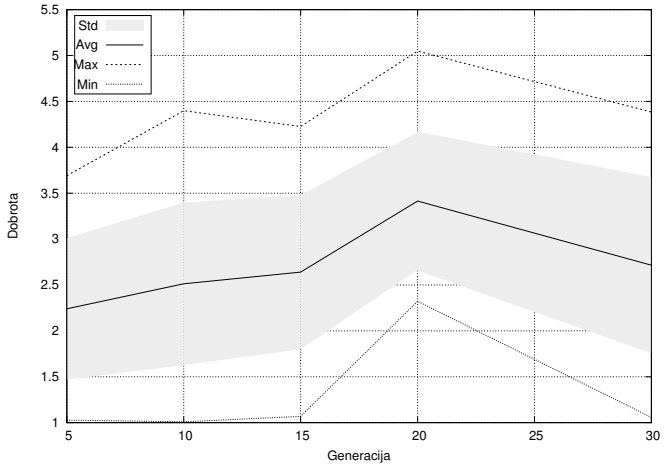
Slika 5.1: Najbolja jedinka (utjecaj evolucijskih parametara)

Tablica 5.4: Vremenski intervali za ispitivanje bez skupa za provjeru

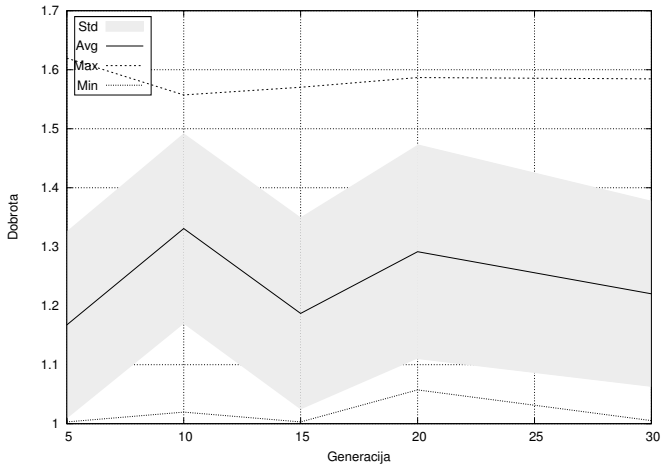
Vremenski period (skup za učenje)	01.01.2000 do 31.12.2007
Vremenski period (skup za ispitivanje)	01.01.2008 do 01.01.2011

5.2.2. Sa skupom za provjeru

Vremenski interval od 01.01.2000 do 31.12.2010 podijeljen je na tri dijela. Vremenski interval skupa za ispitivanje ostao je isti kao i pri ispitivanju bez utjecaja skupa za provjeru. Postojeći vremenski interval skupa za učenje podijeljen je na dva nova skupa, na skup za učenje koji čini oko 75% postojećeg skupa i na skup za provjeru koji koristi preostalih 25% postojećeg skupa. Statistički pregled dobivenih rješenja nalazi se u tablici 5.7. Najbolja jedinka prikazana je na slici 5.9. Dobrota najbolje jedinke na skupu za ispitivanje je $d_i = 1.65818$, a na skupu za provjeru je $d_u = 0.969936$.



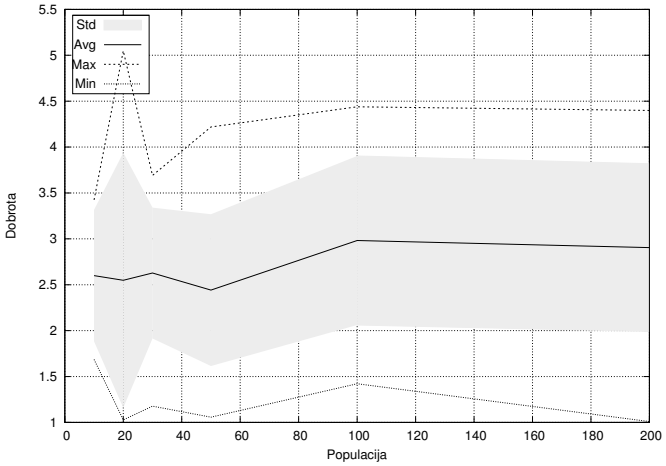
Slika 5.2: Dobrota na skupu za učenje u odnosu na broj generacija



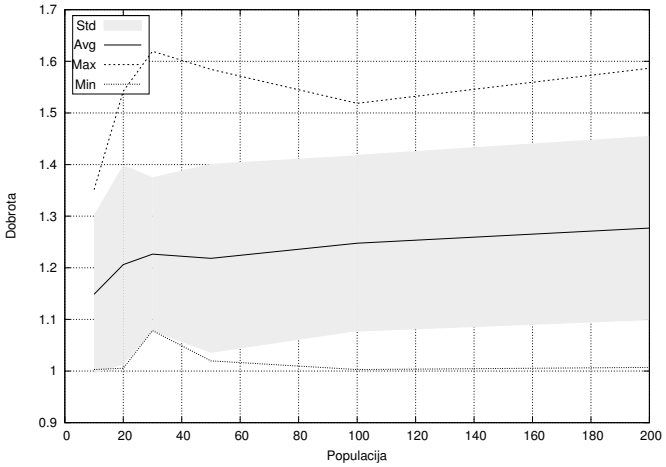
Slika 5.3: Dobrota na skupu za ispitivanje u odnosu na broj generacija

5.3. Usporedba rezultata

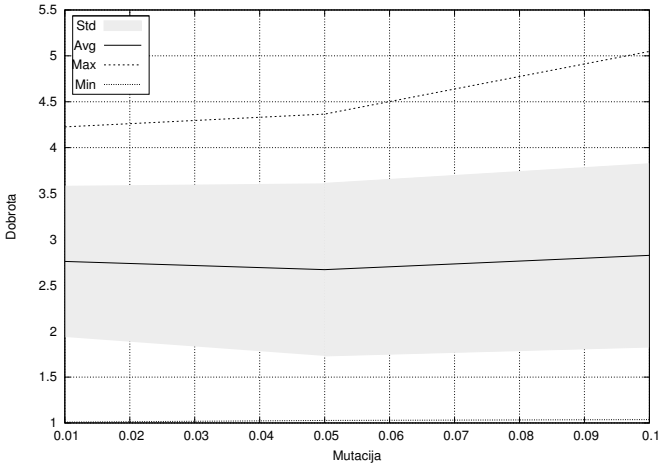
Kako bi se vidio utjecaj skupa za provjeru, jedinke sa (5.2.2) i bez (5.2.1) utjecaja skupa za provjeru ispitane su na vremenskom skupu od 01.01.2008 do 01.01.2011. U tablici 5.8 prikazani su usporedni statistički prikaz dobrota jedinki sa i bez utjecaja skupa za provjeru. Jedinke koje su odabrane na temelju skupa za provjeru u prosjeku su neznatno bolje od onih koje nisu koristile skup za provjeru.



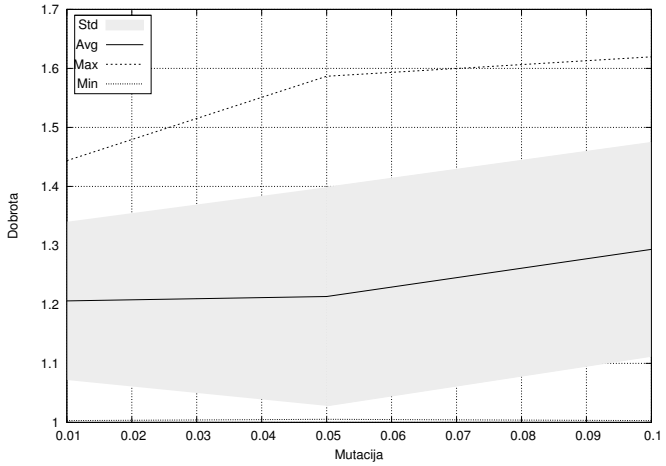
Slika 5.4: Dobrota na skupu za učenje u odnosu na veličinu populacije



Slika 5.5: Dobrota na skupu za ispitivanje u odnosu na veličinu populacije



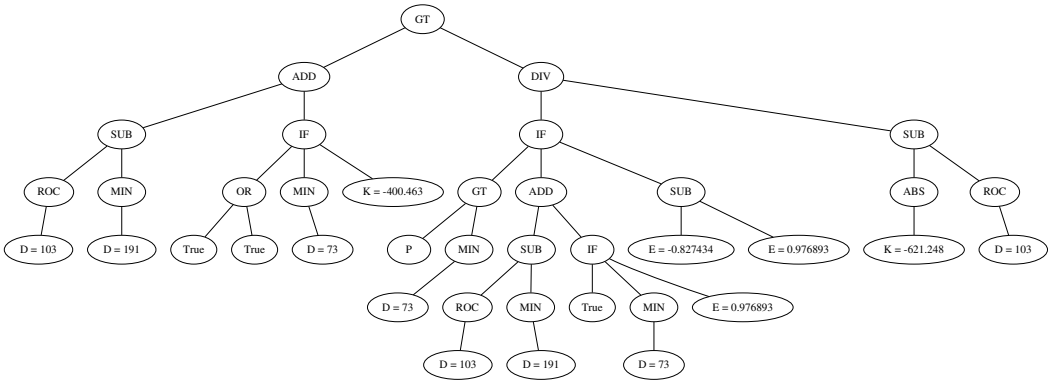
Slika 5.6: Dobrota na skupu za učenje u odnosu na vjerojatnost mutacije



Slika 5.7: Dobrota na skupu za ispitivanje u odnosu na vjerojatnost mutacije

Tablica 5.5: Statistike rezultata bez skupa za provjeru

	Avg	Stddev	Max	Min
Dobrota na skupu za učenje	4.15	1.64	8.28	1.06
Dobrota na skupu za ispitivanje	0.98	0.32	1.59	0.46
Dubina stabla	5.4	2.28	16.0	3.0
Veličina stabla	15.58	12.54	58.0	4.0



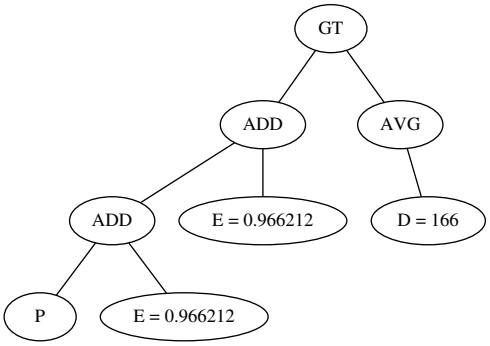
Slika 5.8: Najbolja jedinka bez skupa za provjeru

Tablica 5.6: Vremenski intervali za ispitivnje sa skupom za provjeru

Vremenski period (skup za učenje)	01.01.2000 do 31.12.2005
Vremenski period (skup za provjeru)	01.01.2006 do 31.12.2007
Vremenski period (skup za ispitivanje)	01.01.2008 do 01.01.2011

Tablica 5.7: Statistike rezultata sa skupom za provjeru

	Avg	Stddev	Max	Min
Dobrota na skupu za učenje	4.2	2.44	15.65	1.67
Dobrota na skupu za provjeru	0.77	0.2	1.03	0.38
Dobrota na skupu za ispitivanje	1.01	0.34	1.66	0.48
Dubina stabla	5.47	3.16	15.0	2.0
Veličina stabla	16.18	19.12	90.0	3.0



Slika 5.9: Najbolja jedinka sa skupom za provjeru

Tablica 5.8: Usporedba rezultata uz utjecaj skupa za provjeru

	Avg	Stddev	Max	Min
Dobrota jedinki bez skupa za provjeru (5.2.1)	0.98	0.32	1.59	0.46
Dobrota jedinki uz skup za provjeru (5.2.2)	1.01	0.34	1.66	0.48

6. Zaključak

Genetsko programiranje je moćan alata pomoću kojega možemo uvidjeti zavisnosti u kretanju cijena dionica koje inače ne bi bile očite. Također nizom primjera je pokazano da se strategija „buy and hold“ može nadmašiti pravilom generiranim genetskim programiranjem, ali ipak strategija „buy and hold“ najbolja je kod dionica koje imaju konstantan blagi rast bez puno perturbacija, dok kod onih sa puno skokova i padova dobro naučeni genetski program može ostvariti višestruku dobit. Također promatranjem dobrote na skupu za učenje uočeno je da prilikom velikog broja generacija dolazi do prenaučenosti jedinke. Stoga je poželjno koristiti skup za provjeru te zaustaviti evoluciju kod pojave prenaučenosti.

U nekom od sljedećih istraživanja bilo bi zanimljivo vidjeti kakav utjecaj ima odabrani skup funkcija (čvorova) na konvergenciju rješenja. Naime, skup funkcija (4.3) sadrži funkcije koje su složene od nekih jednostavnijih funkcija koje se također nalaze u skupu. Jedna od takvih funkcija je MACD (2.2.2) koja je definirana kao razlika dvaju EMA (2.2.1). Provjeriti je li konvergencija prema rješenju brža ili sporija upotrebom samo složenih funkcija.

Prilikom kreiranja stabala postoji mogućnost da se pojedini dijelovi stabla nikad ne evaluiraju, jer je npr. uvjet u IF-THEN-ELSE čvoru je uvijek zadovoljen te se njegov inače dio nikad ne evaluira. Iako su takva mjesta pogodna za smještanje rezervnog genetskog materijala, kao i mjesta gdje štetne mutacije nemaju utjecaja na evaluaciju ona uzrokuju povećanje jedinki [1]. Takvi, neaktivni, dijelovi genetskog programa nazivaju se introni. Može se implementirati metoda koja pronalazi takva mjesta i pojednostavnjuje ih, izbacuje IF-THEN-ELSE čvor i zamjenjuje ga samo podstablom koji se izvršava ukoliko je uvjet zadovoljen. Promatra se da li su takva pojednostavnjenja dobra ili loša za konvergenciju prema rješenju.

[1] eng. bloat

LITERATURA

F. Allen i R. Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, 1999.

Lee A. Becker i Mukund Seshadri. Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Technical report, Worcester Polytechnic Institute, Svibanj 2003a. URL `http://citeseer.ist.psu.edu/574013.html`.

Lee A. Becker i Mukund Seshadri. GP-evolved technical trading rules can outperform buy and hold. U *Procceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, Embassy Suites Hotel and Conference Center, Cary, North Carolina USA, Rujan 26-30 2003b. URL `http://www.cs.ucl.ac.uk/staff/W.Yan/gp-evolved-technical-trading.pdf`.

Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, London, 1859. Or the Preservation of Favored Races in the Struggle for Life.

M. A. H. Dempster i C. M. Jones. A real-time adaptive trading system using genetic programming. *Quantitative Finance*, 1:397–413, 2000. URL `http://citeseer.ist.psu.edu/dempster01realtime.html`.

Christian Gagné, Marc Schoenauer, Marc Parizeau, i Marco Tomassini. Genetic programming, validation sets, and parsimony pressure. U Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, i Anikó Ekárt, urednici, *Proceedings of the 9th European Conference on Genetic Programming*, svezak 3905 od *Lecture Notes in Computer Science*, stranice 109–120, Budapest, Hungary, 10 - 12 Travanj 2006. Springer. ISBN 3-540-33143-3. URL `http://link.springer.de/link/service/series/0558/papers/3905/39050109.pdf`.

Wen-Kuei Hsieh i Sung-Yi Hsieh. An application of genetic programming paradigm on the stock market. 2008. URL `http://www.`

aiecon.org/conference/2008/CIEF/An%20Application%20of%20Genetic%20Programming%20Paradigm%20on%20the%20Stock%20Market/An%20Application%20of%20Genetic%20Programming%20Paradigm%20on%20the%20Stock%20Market.pdf.

Efstathios Kalyvas. Using neural networks and genetic algorithms to predict stock market returns. Theisis, 2001.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-11170-5.

T. Kurokawa. Stock trading with genetic algorithm—switching from one stock to another. 2009.

Dome Lohpetch i David Corne. Discovering effective technical trading rules with genetic programming: towards robustly outperforming buy-and-hold. U *World Congress on Nature Biologically Inspired Computing, NaBIC 2009*, stranice 439–444, Prosinac 2009. doi: doi:10.1109/NABIC.2009.5393324.

Dome Lohpetch i David Corne. Outperforming buy-and-hold with evolved technical trading rules: Daily, weekly and monthly trading. U Cecilia Di Chio, Anthony Brabazon, Gianni A. Di Caro, Marc Ebner, Muddassar Farooq, Andreas Fink, Jorn Grahl, Gary Greenfield, Penousal Machado, Michael O'Neill, Ernesto Tarantino, i Neil Urquhart, urednici, *EvoFIN*, svezak 6025 od *LNCS*, stranice 171–181, Istanbul, 7-9 Travanj 2010. Springer. doi: doi:10.1007/978-3-642-12242-2_18.

David J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3 (2):199–230, 1995. doi: doi:10.1162/evco.1995.3.2.199. URL http://vishnu.bbn.com/papers/stgp.pdf.

Nicolas Navet. Genetic Programming for Financial Trading : a Tutorial. U *5th International Conference on Computational Intelligence in Economics and Finance - CIEF 2006*, Kaohsiung Taïwan, Province De Chine, 2006. URL http://hal.inria.fr/inria-00113706/en/. Tutorial given at CIEF'2006 - available at url http://www.loria.fr/ nnavet J.: Computer Applications/J.1: ADMINISTRATIVE DATA PROCESSING/J.1.2: Financial (e.g., EFTS), J.: Computer Applications/J.7: COMPUTERS IN OTHER SYSTEMS.

Christopher J. Neely, Paul A. Weller, i Rob Dittmar. Is technical analysis in the foreign exchange market profitable? A genetic programming approach. *The Jour-*

nal of Financial and Quantitative Analysis, 32(4):405–426, Prosinac 1997. ISSN 00221090. URL http://links.jstor.org/sici?sici=0022-1090%28199712%2932%3A4%3C405%3AITAITF%3E2.0.CO%3B2-T.

A.S. Othling, J.A. Kelly, R.J. Pryor, i G.V. Farnsworth. Successful technical trading agents using genetic programming. Technical Report SAND2004-4774, Sandia National Laboratories, October 1 2004.

Jean-Yves Potvin, Patrick Soriano, i Maxime Vallee. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004. ISSN 0305-0548. doi: doi:10.1016/S0305-0548(03)00063-7. URL http://www.sciencedirect.com/science/article/B6VC5-48GVPS3-1/2/a068a76df94cb8449f6ef7782615fc87.

Christian Setzkorn, Laura Dipietro, i Robin Purshouse. Evolving rule-based trading systems. Technical Report ULCS-02-005, Department of Computer Science, University of Liverpool, UK, 2002. URL http://citeseer.ist.psu.edu/503310.html.

James D Thomas i Katia Sycara. The importance of simplicity and validation in genetic programming for data mining in financial data. U Alex Alves Freitas, urednik, *Data Mining with Evolutionary Algorithms: Research Directions*, stranice 7–11, Orlando, Florida, 18 Srpanj 1999. AAAI Press. ISBN 1-57735-090-1. URL http://citeseer.ist.psu.edu/323257.html. Technical Report WS-99-06.

Liad Wagman. Stock portfolio evaluation: An application of genetic-programming-based technical analysis. U John R. Koza, urednik, *Genetic Algorithms and Genetic Programming at Stanford 2003*, stranice 213–220. Stanford Bookstore, Stanford, California, 94305-3079 USA, 4 Prosinac 2003. URL http://www.genetic-programming.org/sp2003/Wagman.pdf.

Tina Yu, Shu-Heng Chen, i Tzu-Wen Kuo. Discovering financial technical trading rules using genetic programming with lambda abstraction. U Una-May O'Reilly, Tina Yu, Rick L. Riolo, i Bill Worzel, urednici, *Genetic Programming Theory and Practice II*, poglavlje 2, stranice 11–30. Springer, Ann Arbor, 13-15 Svibanj 2004. ISBN 0-387-23253-2.

Dodatak A
Upute za programsku potporu

A.1. Alati korišteni u izradi

Kod izrade ovog diplomskog rada nastojalo se koristiti programsku podršku otvorenog koda (open source) koja je prenosiva na više platformi (prvenstveno razne inačice Linux/Unix operacijskog sustava i Microsoft Windows operacijskog sustava).

- OpenBeagle v3.0.3 je biblioteka funkcija otvorenog koda napisana programskim jezikom C++ te je snažno temeljena na objektnoj orijentiranosti (OO). OpenBeagle je dostupan na: `http://beagle.sourceforge.net/`
- SQLite v3.6.1 je implementacija SQL baze podataka kao biblioteke funkcija. Cijela baza podataka sadržana je u jednoj datoteci koja podržana na svim platformama. SQLite je dostupan na: `http://www.sqlite.org/`
- Gnuplot 4.4 Za crtanje grafova. Gnuplot je dostupan na: `http://www.gnuplot.info/`
- Graphviz 2.26.3 za crtanje stabala generiranih programom. Graphviz je dostupan na: `http://www.graphviz.org/`
- Ruby 1.87, sa dodatnim paketima nokogiri, sqlite3, open-uri i optparse za učitavanje podataka. Ruby je dostupan na: `http:/http://www.ruby-lang.org/`

A.2. Podatci

Programska potpora koristi tablicu ZSE u kojoj se nalaze podaci o pojedinim dionicama. Za učitavanje podataka implementirane su dvije skripte, jedna za učitavanje sa Zagrebačke burze (*import_from_zse.rb*), a druga sa Yahoo! finance portala (*im-*

Listing A.1: SQL Tablica sa podatcima

```
CREATE TABLE ZSE (
    DATUM TEXT,             // Datum trgovanja, (npr. 2008-08-01)
    DIONICA TEXT,           // Oznaka dionice, (npr. AAPL)
    KOLICINA NUMERIC,       // Broj dionica kojima se trgovalo
    NAJNIZA NUMERIC,        // Najniza cijena
    NAJVISA NUMERIC,        // Najvisa cijena
    PROMET NUMERIC,         // Ukupni ostvareni promet
    PROMJENA NUMERIC,       // Promjena od prethodnog dana
    PROSJECNA NUMERIC,      // Prosjecna cijena dionice
    PRVA NUMERIC,           // Cijena dionice na pocetku radnog dana
    ZADNJA NUMERIC,         // Cijena dionice na kraju radnog dana
    PRIMARY KEY(DIONICA,DATUM)
);
```

port_from_yahoo.rb). Skripte se napisane u skriptnom programskom jeziku ruby, te se pokreću na sljedeći način:

```
ruby import-from-yahoo.rb --ticker AAPL
```

Lista parametara i njihovih pretpostavljenih vrijednosti nalazi se u tablici A.1.

Tablica A.1: Parametri skripti za učitavanje podataka

Parametar	*Pretpostavljena vrijednost*	*Opis*
ticker	AAPL	Simbol dionice za koju se preuzimaju podatci.
database	se.db	Ime datoteke u koju se spremaju podatci.
date_start	01.01.1990	Početni datum (u obliku DD.MM.YYYY).
date_end	trenutni	Završni datum (u obliku DD.MM.YYYY).

Određivanje pravila trgovanja dionicama uz pomoć genetskog programiranja

Sažetak

U ovom radu opisano je na koji se način genetsko programiranje može primijeniti na trgovanje na burzi. Genetsko programiranje koristi se za pronalaženje pravila trgovanja dionicama na burzi analizirajući povijesne cijene i volumene trgovanja. Opisano je programsko ostvarenje te rezultati testiranja.

Ključne riječi: genetsko programiranje, evolucijski algoritmi, dionice, trgovanje, pravila trgovanja, tehnička analiza, burza

Determining technical trading rules using genetic programming

Abstract

This paper describes how genetic programming can be applied to stock market trading. Genetic programming is used to find technical trading rules by analyzing historical prices and trading volumes. An implementation is described as well as the testing results.

Keywords: genetic programming, evolution algorithms, stock, trading, trading rules, technical analysis, stock market