Генетичко програмирање

Семинарски рад у оквиру курса
Методологија стручног и научног рада
Математички факултет

Александра Стојановић, Ивана Ивановић,
Александар Стефановић, Оливера Поповић
mi16048@alas.matf.bg.ac.rs, mi16120@alas.matf.bg.ac.rs,
ai16222@alas.matf.bg.ac.rs, mi16064@alas.matf.bg.ac.rs

29. март 2020.

Абстракт

U ovom tekstu je ukratko prikazana osnovna forma seminarskog rada. Obratite pažnju da je pored ove .pdf datoteke, u prilogu i odgovarajuća .tex datoteka, kao i .bib datoteka korišćena za generisanje literature. Na prvoj strani seminarskog rada su naslov, apstrakt i sadržaj, i to sve mora da stane na prvu stranu! Kako bi Vaš seminarski zadovoljio standarde i očekivanja, koristite uputstva i materijale sa predavanja na temu pisanja seminarskih radova. Ovo je samo šablon koji se odnosi na fizički izgled seminarskog rada (šablon koji *morate* da koristite!) kao i par tehničkih pomoćnih uputstava. Pročitajte tekst pažljivo jer on sadrži i važne informacije vezane za zahteve obima i karakteristika seminarskog rada.

Садржај

1 Увод **3**

2 Историјат **3**

3 Опис методе **3**
3.1 Општи алгоритам 3
3.1.1 Репрезентација јединки 4
3.1.2 Иницијализација популације 5
3.1.3 Функција прилагођености 6
3.1.4 Селекција 6
3.1.5 Репродукција 6
3.1.6 Мутација 7

4 Примери примене **7**
4.1 Гејминг 7
4.2 Роботика 7
4.3 Медицина 8
4.4 Економија 9

5 Мета-генетичко програмирање **9**

6 Osnovna uputstva **10**

7 Engleski termini i citiranje **10**

8 Slike i tabele **11**

9 Kôd i paket listings **11**

10 Закључак **12**

Literatura **12**

A Dodatak **13**

1 Увод

Kada budete predavali seminarski rad, imenujete datoteke tako da sadrže redni broj teme, temu seminarskog rada, kao i prezimena članova grupe. Precizna uputstva na temu imenovnja će biti data na formi za predaju seminarskog rada. Predaja seminarskih radova biće isključivo preko veb forme, a NE slanjem mejla. Link na formu će biti dat u okviru obaveštenja na strani kursa. Vodite računa da prilikom predavanja seminarskog rada predate samo one fajlove koji su neophodni za ponovno generisanje pdf datoteke. To znači da pomoćne fajlove, kao što su .log, .out, .blg, .toc, .aux i slično, **ne treba predavati**.

2 Историјат

3 Опис методе

Генетичко програмирање је метода која је настала као један од покушаја да се спроведе аутоматизација прављења рачунарских програма за решавања проблема задатих на веома апстрактном нивоу. Основни циљ је да ови проблеми буду тако дефинисани да је познато једино шта треба да буде урађено, без додатних информација о форми или структури решења.

Идеја на којој ова метода почива је процес еволуције који се јавља у природи. Самим тим, генетичко програмирање је техника која спада у групу алгоритама **еволутивног израчунавања** [3]. Еволуција подразумева промену особина укупне популације кроз више смењених генерација. Постојало је током времена више теорија еволуције, док је данас опште прихваћена теорија чији је творац британски биолог Чарлс Роберт Дарвин. По његовој теорији, сви живи организми се развијају процесом природне селекције [2]. Тим процесом јединке које су боље прилагођене имају већу вероватноћу да преживе и да оставе потомство, које је углавном једнако или чак и боље прилагођено од родитеља. Да би могли да оставе потомство, у ћелијама сваког живог бића постоје хромозоми, који су сачињени од гена. Ген је основна јединица наслеђивања, која преноси поруке из генерације у генерацију. Дакле, репродукција подразумева комбинацију гена родитеља уз мале количине мутације. Карактеристике, односно генетски материјал прилагођених јединки углавном опстаје кроз генерације и у неком тренутку почиње да доминира, док остале особине најчешће ишчезну.

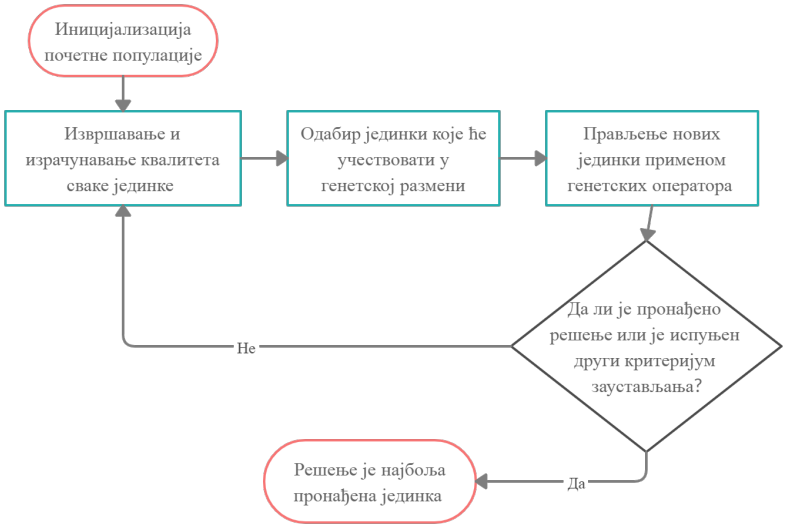
У овом одељку биће описано како се процес еволуције осликава на генетичко програмирање, које су његове основне компоненте као и то које одлуке треба донети при прилагођавању алгоритма задатом проблему.

3.1 Општи алгоритам

У генетичком програмирању популацију чине рачунарски програми. Еволуцијом се од почетних, најчешће на случајан начин генерисаних програма, добијају нови, у нади бољи и ефикаснији програми. Обзиром да је то случајан процес, резултат никада не може бити гаранто-

ван. Ипак, управо та случајност даје могућност превазилажења неких проблема које имају други детерминистички алгоритми [10].

На слици 1 приказан је дијаграм контроле тока опште верзије алгоритма. У наредним поглављима биће детаљно описан сваки од наведених корака алгоритма.



Слика 1: Општи алгоритам генетичког програмирања

3.1.1 Репрезентација јединки

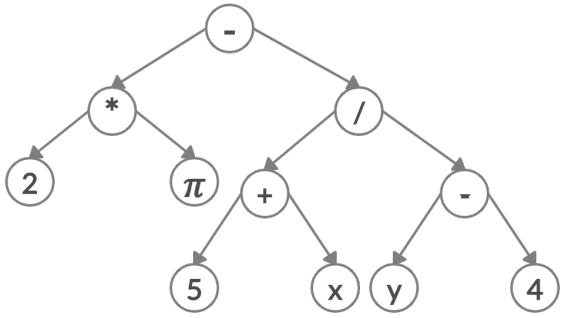
У генетичком програмирању, јединке су заправо рачунарски програми који се могу извршавати. Ипак, иако то на први поглед делује логично, оне неће бити представљене линијама кода. Разлог томе је што је неопходно да над изабраном репрезентацијом буду дефинисани генетски оператори (више о њима у наредним секцијама 3.1.5 и 3.1.6), а да извршавање буде ефикасно. Репрезентација која то испуњава је она у виду **синтаксних стабала** [1], те се она најчешће користи.

Код синтаксних стабала, основно је да се промењиве и константе налазе у листовима (најнижи чворови) и називају се терминалима, док се у осталим чворовима налазе функције. Функције могу бити различите природе, на пример аритметичке, тригонометријске или логичке, али могу бити и различите програмске конструкције попут условних израза и петљи. Свако овакво стабло мора имати скуп дефинисаних терминала и функција који заједно чине такозвани **примитивни скуп** генетичког програмирања.

Пример 3.1 *Нека је проблем задат аритметичком формулом:*

$$2 * \pi - \frac{5 + x}{y - 4} \tag{1}$$

*Одговарајући скуп терминала: {2, π, 5, x, y, 4}, скуп функција: {+, -, *, /}, док је одговарајуће синтаксно стабло дато на слици 2.*



Слика 2: Синтаксно стабло које одговара формули (1)

Репрезентација стаблима има неколико импликација које треба имати на уму [3]:

- Величина (висина), облик (какве су гране чворова) и сложеност јединки нису фиксирани, већ се могу разликовати. Те особине се чак и код једне јединке такође могу мењати током времена, услед примене оператора за репродукцију.
- За сваки проблем се посебно мора дефинисати граматика, односно мора се дефинисати примитиван скуп тако да је њиме могуће представити сва могућа решења. Као додатак примитивном скупу, могу бити дефинисана и семантичка правила која обезбеђују конструкцију исправних стабала. На пример, једно такво правило могло би да буде да делилац никада не сме да буде једнак 0.

Некада се може испоставити да је репрезентација стаблима недовољно удобна за рад јер захтева управљање великим бројем показивача. Најчешће велика већина функција које се користе има фиксирану арност (број аргумената), што омогућава да се стабла прикажу као линеарне структуре (листе, низови). Која ће се репрезентација користити зависи највише од домена проблема који се решава [10].

3.1.2 Иницијализација популације

Генерисање почетне популације обично се врши на случајан начин, при чему се поштују сва дефинисана ограничења, уколико она постоје. Две најстарије и најједноставније методе су **потпуна метода** (eng. *full method*) и **метода раста** (eng. *grow method*), као и њихова широко распрострањена пола-пола комбинација [10]. Да би се разумеле ове две методе, потребно је прво увести две дефиниције:

- **Дубина чвора** представља број грана које треба прећи да би се од корена дошло до тог чвора. Корен се подразумевано налази на дубини 0.
  Дубина чвора са вредношћу π приказаног на слици 2 износи 2.
- **Дубина читавог стабла** је дубина његовог најдубљег листа.
  Дубина стабла приказаног на слици 2 износи 3.

Прва метода зове се потпуна јер креира потпуна стабла. Стабло је потпуно уколико су му сви листови на истој дубини. По овој методи,

чворови се бирају на случајан начин све док се не дође до максималне дубине. Након тога се бирају на случајан начин из скупа терминала. По другој методи, методи раста, чворови се бирају на случајан начин из целог примитивног скупа док се не дође до максималне дубине, а након тога се исто бирају само терминали. Основна разлика између ове две методе је што метода раста дозвољава креацију стабала са више различитих величина и облика.

Ипак, ниједна од ове две методе не пружа претерано велику разноврсност. Због тога се данас нашироко користи њихова комбинација коју је предложио Џон Коза. Она подразумева да се половина генерације креира коришћењем потпуне методе, а друга половина методом раста.

3.1.3 Функција прилагођености

Функција прилагођености (eng. *fitness function*) даје оцену **квалитета јединке**. Она треба да буде одабрана тако да може да се израчуна за сваку јединку а да израчунавање притом буде што ефикасније [6]. Зависи од проблема који се решава, али се у генетичком програмирању најчешће реализује тако што постоји скуп тест случајева којима се свака јединка евалуира. То значи да постоји одређени број улаза за које се знају очекивани излази, па се евалуира колико излаза је јединка, односно програм погодио. Осим као мера квалитета јединке, функција прилагођености се такође може користити да додели пенале оним јединкама које нису прихватљиве или нису семантички тачне.

3.1.4 Селекција

Селекција се користи како би се пронашле боље прилагођене јединке које ће учествовати у репродукцији. То се врши коришћењем функције прилагођености. На основу њене вредности јединке се узимају са одређеном вероватноћом. Најчешће се користе **турнирска селекција** и селекција пропорционала вредности функције прилагођености, али се могу користити и друге селекционе методе еволутивног израчунавања. Код турнирске селекције бира се одређени број јединки из популације, на случајан начин. Оне се упоређују и бира се најбоља од њих, која ће учествовати у репродукцији. Постоје и разне друге варијанте турнирске селекције, али је њена предност свакако то што и средње квалитетне јединке могу добити шансу да оставе потомке. Пример селекције пропроционалне вредности функције прилагођености је **рулетска селекција** (eng. *roulette wheel selection*). По њој, вероватноћа да ће јединка бити изабрана једнака је:

$$p_i = \frac{f(i)}{\sum_j^N f(i)} \tag{2}$$

где је f(i) вредност функције прилагођености јединке i, док је N укупан број јединки у популацији [6].

3.1.5 Репродукција

Репродукција представља начин на који се од два селекцијом изабрана родитеља добиј потомство. Код генетичког програмирања најчешће се користи метод укрштања подстабала. На случајан начин

одаберу се тачке укрштања код оба родитеља, након чега се једноставно замене подстабла родитеља са кореном у тачки укрштања. Постоје две опције, креирање једног или два детета разменом. Проблем код овог метода је то што се најчешће барата са стаблима код којих сваки чвор има најмање двоје деце, што доводи до тога да већи број чворова припада скупу терминала. Последица тога је што се размењују мање количине генетског материјала, јер се чешће на случајан начин погоди управо неки од терминала. Да би се избегао овај проблем, Коза је 1992. предложио нашироко коришћен приступ бирања функција 90% времена, а листова преосталих 10%. Постоји још много видова репродукције који се користе код генетичког програмирања [10].

3.1.6 Мутација

Мутација се примењује како током времена јединке не би постале сувише сличне. Такође помаже у избегавању локалних екстремума, у ком би се највероватније јединке заглавиле уколико не би било мутације. Мутација се најчешће имплементира тако да се на случајан начин изабере тачка јединке, па се подстабло са кореном у тој тачки замени случајно генерисаним стаблом. Још један чест начин имплементације је да се такође на случајан начин изабере тачка у стаблу јединке, али се сада вредност у том чвору замени случајно одабраним

4 Примери примене

Примена генетичког програмирања у индустрији је разнолико и рапидно се повећава годинама. Споменућемо неке од важнијих радова написаних о изабраним темама.

4.1 Гејминг

4.2 Роботика

Један од проблема у роботици укључује два агента А и Б који управљају својим возилима на дводимензионалној мрежи. Возила се све време померају у правцу у коме је возило окренуто. Сви покрети су дискретни, тако да возила имају увек целобројне (x, y) координате. Иницијално возила су окренута ка северу, а механизам управљања возилом дозвољава окретање возила у једном од 8 географских праваца. Циљ је да агенти А и Б својим возилима дођу један до другог.

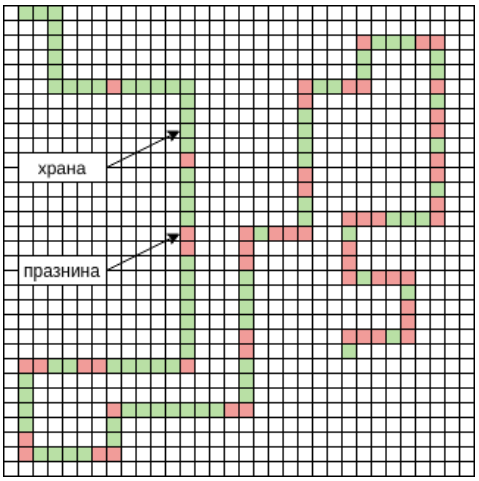
Агенти А и Б имају информацију само о сопственој позицији, али такође имају приступ каналима за комуникацију КаналА и КаналБ који се могу користити за слање и примање порука од другог агента. Сваки комуникациони канал је конектован са једним возилом, тако да порука послата са једног краја може бити примљена само на другом крају и обрнуто. [11]

Сличан овоме, проблем пре тога укључивао је два играча у возилима који су се играли јурке. С тим што је овде сваки играч имао информацију о томе ко јури и где му се апроксимативно налази противник. Циљ је што краће бити онај који треба да јури противника. Играју се четири игре, смењујући се у свакој рунди ко први јури. У току једне рунде улоге се мењају кад онај који јури буде од свог противника на дужини од једног возила. Свака рунда се састоји из

25 симулација, где је играчев резултат за ту рунду рачунат као број колико пута није био онај који јури подељен са 25. [12]

Такође, постоји и проблем навигације робота како би нашао храну која је постављена дуж неправилне линије; где робот зна да извршава примитивне операције: померање напред, окретање лево/десно као и мирисање хране. Путање које су коришћене су:

- "Santa Fe" путања (приказана на слици 3) - направљена од 89 комада хране, где између њих постоји једна, две или три празнине без хране који могу бити и на угловима линије кретања робота
- "Los Altos Hills" путања - састављена од 157 комада хране, где 105. комад представља 89. комад из претходне путање. Након тог комада, додате су још две ирегуларности у путањи:
  - налажење хране два места лево или десно од претходног комада хране; прво појављивање после 116. комада хране
  - померање једно место унапред, а онда проналажење хране два месте лево или десно; прво појављивање после 136. комада хране



Слика 3: Изглед "Santa Fe" путање

Успешно је било проналажење све хране од стране робота у 21. генерацији за прву путању, а у 19. генерацији за другу путању. [7] Овај проблем је решен уз помоћ програмског језика Lisp и коришћењем с-израза. [9]

4.3 Медицина

Иако генетичко програмирање не може бити једино решење када се анализирају биолошки подаци, оно је важно као додатан аналитички алат који има потенцијал да пружи изненађујуће и непредвидиве резултате. Способност да пронађе корисне комбинације својстава и да их прикаже у облику разумљивом за људе га чине погодним за истраживање болести попут рака. [16]

Коришћење унапређеног генетичког програмирања је показало боље резултате у проблему класификације података од класичних алгоритама машинског учења попут линеарне регресије, неуронских мрежа и класификације базиране на суседима. [15]

4.4 Економија

Можда једна од најбитнијих проблема са којима се сусреће ова дисциплина је предвиђање банкрупта компанија. Коришћењем генетичког програмирања дошло се до занимљивих карактеристика података ових предузећа. [13]

Такође, у економији је од велике важности предвидети податке на тржиштима хартија од вредности (берзама), како због смањења губитака од куповине и продаје акција, тако и зарад максимизовања профита. [5]

5 Мета-генетичко програмирање

Мета-генетичко програмирање подразумева упоредно еволуирање **(да ли је добар термин?)** програма који врши генетичко програмирање, које замењује ручно подешавање параметара и одлика алгоритма.

citat There is no reason to believe that having only one level of adaption in an evolutionary computation is optimal. Обично су оператори генетичког програмирања унапред задати и фиксни, и најчешће се ради о комбинацији **propagation, crossover, mutation**, али су истраживани и неки алтернативни оператори **referenca?**, и испоставља се да у неким случајевима, коришћење тих алтернативних оператора, заједно са „класичним" операторима даје боље резултате. Дакле, паметним одабиром и укључивањем одређених оператора мутације, могуће је постићи већи учинак програма који врши генетичко програмирање.

Међутим, одабир оператора није тривијалан, јер је могуће да неки јако добар оператор не буде ни лак за тумачење, ни интуитиван, односно, мало је вероватно да би нека особа сама осмислила такав оператор, што сасвим одговара мотивацији генетичких алгоритама уопште, где је простор претраге јако велик, и посао проналажења решења је препуштен рачунару, а не особи.

Као и у типичном генетичком програмирању, јединке **geni?** се приказују помоћу синтаксних стабала, с тим што оваква стабла садрже инструкције које представљају операторе мутације.

fitness funkcija узима у обзир могућност добијеног програма да ефикасно и ефективно еволуира програме, што укратко значи да је **fitness funkcija** самог програма који врши генетичко програмирање донекле сразмерна и **фитнес функцији** јединки које генерише.

6 Osnovna uputstva

Vaš seminarski rad mora da sadrži najmanje jednu **sliku**, najmanje jednu **tabelu** i najmanje **sedam referenci** u spisku literature. Najmanje jedna slika treba da bude originalna i da predstavlja neke podatke koje ste Vi osmislili da treba da prezentujete u svom radu. Isto važi i za najmanje jednu tabelu. Od referenci, neophodno je imati bar jednu **knjigu**, bar jedan **naučni članak** iz odgovarajućeg časopisa i bar jednu adekvatnu **veb adresu**.

Dužina seminarskog rada treba da bude od 10 do 12 strana. Svako prekoračenje ili potkoračenje biće kažnjeno sa odgovarajućim brojem poena. Eventualno, nakon strane 12, može se javiti samo tekst poglavlja **Dodatak** koji sadrži nekakav dodatni kôd, ali je svakako potrebno da rad može da se pročita i razume i bez čitanja tog dodatka.

Ко жели, може да пише рад ћирилицом. У том случају, неопходно је да су инсталирани одговарајући пакети: texlive-fonts-extra, texlive-latex-extra, texlive-lang-cyrillic, texlive-lang-other.

Nemojte koristiti stari način pisanja slova, tj ovo:

```
\v{s} i \v{c} i \'c ...
```

Koristite direknto naša slova:

```
š i č i ć ...
```

7 Engleski termini i citiranje

Na svakom mestu u tekstu naglasiti odakle tačno potiču informacije. Uz sve novouvedene termine u zagradi naglasiti od koje engleske reči termin potiče.

Naredni primeri ilustruju način uvođenja enlegskih termina kao i citiranje.

Пример 7.1 *Problem zaustavljanja (eng.* halting problem*) je neodlučiv [14].*

Пример 7.2 *Za prevođenje programa napisanih u programskom jeziku C može se koristiti GCC kompajler [4].*

Пример 7.3 *Da bi se ispitivala ispravost softvera, najpre je potrebno precizno definisati njegovo ponašanje [8].*

Reference koje se koriste u ovom tekstu zadate su u datoteci *seminarski.bib*. Prevođenje u pdf format u Linux okruženju može se uraditi na sledeći način:

```
pdflatex TemaImePrezime.tex
bibtex TemaImePrezime.aux
pdflatex TemaImePrezime.tex
pdflatex TemaImePrezime.tex
```

Prvo latexovanje je neophodno da bi se generisao *.aux* fajl. *bibtex* proizvodi odgovarajući *.bbl* fajl koji se koristi za generisanje literature. Potrebna su dva prolaza (dva puta pdflatex) da bi se reference ubacile u tekst (tj da ne bi ostali znakovi pitanja umesto referenci). Dodavanjem novih referenci potrebno je ponoviti ceo postupak.

Broj naslova i podnaslova je proizvoljan. Neophodni su samo Uvod i Zaključak. Na poglavlja unutar teksta referisati se po potrebi.

Пример 7.4

Još jednom da napomenem da nema razloga da pišete:

`\v{s} i \v{c} i \'c ...`

Možete koristiti srpska slova

`š i č i ć ...`

8 Slike i tabele

Slike i tabele treba da budu u svom okruženju, sa odgovarajućim naslovima, obeležene labelom da koje omogućava referenciranje.

Пример 8.1 *Ovako se ubacuje slika. Obratiti pažnju da je dodato i* `\usepackage{graphicx}`

Слика 4: Pande

Na svaku sliku neophodno je referisati se negde u tekstu. Na primer, na slici 4 prikazane su pande.

Пример 8.2 *I tabele treba da budu u svom okruženju, i na njih je neophodno referisati se u tekstu. Na primer, u tabeli 1 su prikazana različita poravnanja u tabelama.*

Табела 1: Razlčita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

9 Kôd i paket listings

Za ubacivanje koda koristite paket **listings**: https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings

Пример 9.1 *Primer ubacivanja koda za programski jezik Python dat je kroz listing 1. Za neki drugi programski jezik, treba podesiti odgvarajući programski jezik u okviru defnisanja stila.*

```
# This program adds up integers in the command line
import sys
try:
    total = sum(int(arg) for arg in sys.argv[1:])
    print 'sum =', total
except ValueError:
    print 'Please supply integer arguments'
```

Listing 1: Primer ubacivanja koda u tekst

10 Закључак

Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak. Ovde pišem zaključak.

Литература

[1] A Aho, M.S. Lam, R. Sethi, and J. Ullman. *Compilers: principles, techniques and tools*. Addison-Wesley Longman Publishing Co., 2006.

[2] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, London, 1859.

[3] A.P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, Ltd, 2007.

[4] Free Software Foundation. GNU gcc, 2013. on-line at: http://gcc.gnu.org/.

[5] Sasaki T. Iba H. Using genetic programming to predict financial data. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, pages 244–251, july 1999.

[6] Predrag Janičić and Mladen. Nikolić. *Vestačka inteligencija*. Matematički fakultet u Beogradu, 2019.

[7] John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, London, 1992.

[8] J. Laski and W. Stanley. *Software Verification and Analysis*. Springer-Verlag, London, 2009.

[9] John McCarthy. Lisp, 1958. on-line at: https://lisp-lang.org/.

[10] R Poli, W.B. Langdon, and N.F. McPhee. *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[11] A. Qureshi. Evolving agents. *University College of London*, 1996.

[12] Craig W. Reynolds. Competition, Coevolution and the Game of Tag. In *Proceedings of Artificial Life IV*, pages 59–69, jan 1994.

[13] Terje Lensberg Thomas E. McKee. Genetic Programming and rough sets: A hybrid approach to bankruptcy classification. *European Journal of Operational Research*, 1(138):436–451, 2002.

[14] A. M. Turing. On Computable Numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[15] Stephan Winkler, Michael Affenzeller, and Stefan Wagner. Using enhanced genetic programming techniques for evolving classifiers in the context of medical diagnosis. *Genetic Programming and Evolvable Machines*, 10:111–140, 06 2009.

[16] William Worzel, Jianjun Yu, Arpit Almal, and Arul Chinnaiyan. Applications of genetic programming in cancer research. *The International Journal of Biochemistry & Cell Biology*, 41:405–413, 02 2009.

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.