SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4833

PRONALAŽENJE STRATEGIJE U IGRAMA S VIŠE IGRAČA UZ POMOĆ EVOLUCIJSKIH ALGORITAMA

Ivona Škorjanc

Zagreb, lipanj 2017.

Sadržaj

1. Uvod

Sve popularnije područje umjetne inteligencije može dati rješenja za iznimno širok spektar problema. Ovaj rad fokusirat će se na problem pronalaženja strategije za ponovljene igre u više igrača. Ponovljenim igrama mogu se simulirati razne situacije odlučivanja i ponašajne značajke čovjeka. Problem pronalaženja strategije je zanimljiv jer nije sasvim intuitivno niti jednostavno smisliti dobru strategiju za neku od takvih igara.

Kao alat za pronalaženje strategija koristit ćemo jednu od grana strojnog učenja – evolucijske algoritme. Postoji više prikladnih modela rješavanja problema pomoću evolucijskih algoritama. Implementacijsko ostvarenje bit će detaljnije objašnjeno na jednom od modela. Izvođenje i testiranje provodit će se na dvije ponovljene igre: Zatvorenikova dilema i Kamen-škare-papir.

2. Ponovljene igre za više igrača

Koncept ponovljenih igara koriste teoretičari igara, ekonomisti, sociolozi i drugi znanstvenici kao pojednostavljene modele raznih svakodnevnih društvenih situacija. Radi se o ponavljanju dobro poznate igre, najčešće u dva igrača, veći broj puta. Taj broj može biti poznat ili nepoznat, konačan ili beskonačan.

Po čemu se ponovljene igre razlikuju od igara koje se igraju samo jednom? Igrajući jedan krug, igrači će uvijek odabrati onaj potez koji će im trenutno donijeti najveću dobit ili najmanji gubitak. Igrajući više iteracija, dobitak/gubitak se akumulira i više nije tako jednostavno odrediti koji je dugoročno najisplativiji potez. Postoji puno veći prostor mogućih **strategija**.

Najčešći primjer koji se u teoriji igara uzima za objašnjavanje ponovljenih igara je Iterirana zatvorenikova dilema.

2.1 Iterirana zatvorenikova dilema

Zatvorenikova dilema je igra s dva igrača: dvoje ljudi uhićeno zbog sumnje da su počinili zločin. Pritvorenici se ispituju zasebno i ne znaju što će reći onaj drugi. Svaki ima dva izbora: surađivati s drugim (ne reći ništa policiji) ili ga izdati. Pravila su sljedeća:

- Maksimalna kazna je 10 godina zatvora – ukoliko prvi igrač surađuje a drugi ga izda, prvi dobiva 10 godina a drugi je slobodan (i obrnuto)
    - Vrlo loše za prvoga (0 bodova), vrlo dobro za drugoga (5 bodova)
- Ako oba igrača surađuju dobivaju po 2 godine zatvora
    - Relativno dobro za oboje (3 boda)
- Ako oba igrača izdaju drugoga dobivaju po 5 godina zatvora
    - Relativno loše za oboje (1 bod)

Tablica 2.1 Zatvorenikova dilema

igrač 1, igrač 2	suradnja	izdaja
suradnja	3, 3	0, 5
izdaja	5, 0	1, 1

Gledajući svakog igrača zasebno, povlačenje poteza „izdaja“ uvijek će biti bolje od povlačenja poteza „suradnja“ (5>3 i 1>0). No, scenarij u kojem oba igrača povuku taj potez zapravo je ukupno najlošiji (donosi najmanje bodova – samo 2). Kad se igra samo jednom, igrači će uvijek najbolje proći ako izdaju drugoga. Ako se igra više puta i rezultati se sumiraju, strategija se može promijeniti. U ovom slučaju „izdaja“ nosi posljedice i stalno povlačenje tog poteza neće igraču donijeti maksimalni broj bodova.

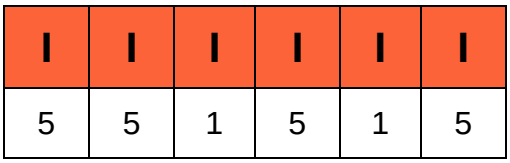
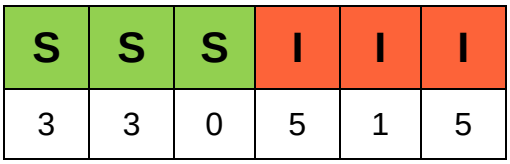
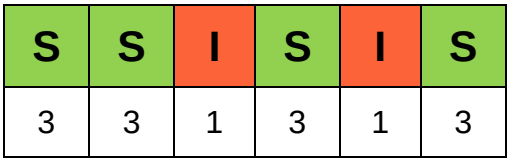
Pojasnimo to malo detaljnije na jednom primjeru. Uzmimo strategiju koja povlači nasumične poteze, bez obzira na to što je protivnik odigrao. Neka taj niz bude slijedeći (S = suradnja, I = izdaja):

S	S	I	S	I	S

Pogledajmo rezultate u igri protiv nekih strategija:

Tablica 2.2 Simulacija igre bez posljedica za „izdaju“

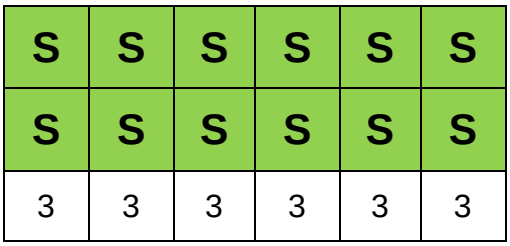
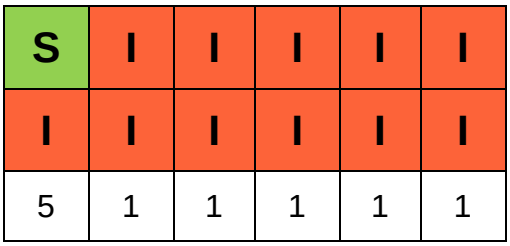
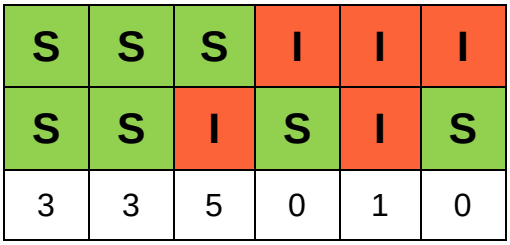
strategija		broj bodova
	S S I S I S	
samo suradnja	S S S S S S 3 3 0 3 0 3	12

samo izdaja		**22**
prvo suradnja zatim izdaja		17
kopiranje		15

Vidimo da stalno povlačenje poteza „izdaja“ donosi najviše bodova. To se događa jer je strategija fiksna – nema posljedica povlačenja određenog poteza. Ovakva igra se ne razlikuje od igranja samo jednom.

Pogledajmo sada što će se dogoditi ako igrač bira svoj idući potez na osnovu onoga što je povukao njegov protivnik. Uzmimo strategiju koja uvijek surađuje, sve dok protivnik ne povuče potez „izdaja“. Strategija će kazniti protivnika time što će svaki idući potez do kraja igre biti „izdaja“ (protivnik više ne može osvojiti niti 3 niti 5 bodova).

Tablica 2.3 Simulacija igre s posljedicama za „izdaju“

samo suradnja		**18**
samo izdaja		10
pokušaj suradnje nakon izdaje		12

Na primjeru vidimo da su posljedice za „izdaju“ sve promijenile. Sada se najviše isplati surađivati. Ovaj princip pokazao se točan i za ostale strategije.

Opisana igra gdje se Zatvorenikova dilema primjenjuje kroz više uzastopnih krugova naziva se Iterirana zatvorenikova dilema [1]. Teorija ove igre prvi put je opisana u knjizi „The Selfish Gene“ Richarda Dawkinsa. Često se koristi za objašnjavanje evolucije kooperativnog ponašanja. Politolog Robert Axlerod održao je prvi turnir [2] s ciljem pronalaženja najbolje strategije koja će sakupiti najviše bodova u igri. Rezultate je iskoristio u svojoj knjizi Evolucija kooperacije. Nakon prvoga održano je još nekoliko turnira.

Razne strategije su se na turnirima pokazale više ili manje učinkovitima. No, najistaknutiji zaključak je da kooperativne strategije daju daleko bolje rezultate od nekooperativnih.

Generalno, kooperativne strategije su one koje prve povlače potez „suradnja“ i nastoje što više surađivati kroz igru. Ukoliko protivnik povuče potez „izdaja“, ove strategije će ga najčešće na neki način kazniti (također povući potez „izdaja“). Nekooperativne strategije su one koje će pokušati sakupiti što više bodova nauštrb protivnika.

Jedna strategija koja se na prvih nekoliko turnira izdvojila kao najbolja zove se „Tit for Tat“. Ideja je vrlo jednostavna: povući isti potez koji je protivnik povukao u prošlom potezu. Tablica 2.4 predstavlja pregled nekoliko jednostavnijih strategija koje su se pojavile na turnirima:

Tablica 2.4 Neke od strategija za igru Iterirana zatvorenikova dilema

Kooperativne strategije	Opis
Always cooperate	Uvijek povlači potez „suradnja“
Tit for Tat	Ponavlja zadnji potez protivnika, počinje suradnjom
Tit for two Tats	Surađuje, povlači potez „izdaja“ tek nakon što je protivnik dvaput zaredom povukao taj potez
Grudger	Surađuje dok protivnik ne povuče potez „izdaja“, zatim stalno povlači potez „izdaja“
Pavlov	Ponavlja svoj zadnji potez ako je imao dobar ishod (dobio 3 ili 5 bodova), započinje suradnjom
Neutralne strategije	
Random	Povlači nasumično odabran potez
Alternating moves	Prvi potez bira nasumično, svaki idući povlači suprotan potez

Nekooperativne strategije	
Always defect	Uvijek povlači potez „izdaja“
Suspicious Tit for Tat	Kao Tit for Tat, ali počinje „izdajom“

Često se ovim strategijama doda i element nasumičnosti. Npr, strategija Tit for Tat može s nekom malom vjerojatnošću povući potez „izdaja“ iako je protivnik prethodno surađivao.

2.2 Ostali primjeri

Postoji još primjera igara koje se mogu promatrati u kontekstu ponovljenih igara. Jedna od njih je i vrlo poznata Kamen-škare-papir. Malo je zahtjevnija od Iterirane zatvorenikove dileme jer ima 3 poteza, i nije toliko intuitivno interpretirati razne strategije. Bodovanje se može ostvariti na razne načine, na primjer:

Tablica 2.5 Bodovanje u igri Kamen-škare-papir

igrač 1, igrač 2	Kamen	Škare	Papir
Kamen	1, 1	3, 0	0, 3
Škare	0, 3	1, 1	3, 0
Papir	3, 0	0, 3	1, 1

3. Evolucijski algoritmi i ECF

Evolucijski algoritmi inspirirani su prirodom. Probleme rješavanju metodom pokušaja i pogrešaka na način da se populacije rješenja pokušavaju optimizirati kroz velik broj generacija. Generacije se mijenjaju i napreduju korištenjem operatora selekcije, križanja i mutacije. Za ocjenjivanje rješenja zadužena je funkcija dobrote. Budući da nije potrebno konkretno znanje o problemu ili znanstvenom području, evolucijski algoritmi mogu se primijeniti na širok spektar problema.

```
Evolucijski_algoritam
t = 0
generiraj početnu populaciju potencijalnih rješenja P(0);
sve dok nije zadovoljen uvjet završetka evolucijskog procesa {
     t = t + 1;
     selektiraj P'(t) iz P(t-1);
     križaj jedinke iz P'(t) i djecu spremi u P(t);
     mutiraj jedinke iz P(t);
}
ispiši rješenje;
```

ECF (Evolutionary Computation Framework) [3] je C++ razvojno okruženje namijenjeno primjeni evolucijskog programiranja. Sadrži velik broj algoritama, genotipova (načina prikaza rješenja), operatora mutacije i križanja i dodatnih mogućnosti.

Za svaki problem koji se rješava pomoću ECF-a može se odabrati različit algoritam. Evolucijski algoritam koji je zadan u slučaju da se eksplicitno ne navede neki drugi zove se „Steady State Tournament“ (turnir sa stalnim stanjem). Njega ćemo koristiti u ovom radu.

```
Steady_State_Tournament
ponavljaj (za svaku generaciju){
      nasumce dodaj n jedinki u turnir;
      selektiraj najlošiju jedinku u turniru;
      nasumce odaberi dva roditelja od preostalih jedinki;
      zamijeni najgoru jedinku djetetom dobivenim križanjem
      roditelja;
      mutiraj dijete;
}
```

4. Traženje optimalne strategije igranja ponovljenih igara evolucijskim algoritmima

4.1 Postojeća rješenja

Razni se znanstveni radovi bave tematikom traženja strategije igara u više igrača uz pomoć evolucijskih algoritama. Koriste se različiti pristupi koji se primjenjuju na različitim igrama.

U radu [4] odabrana je igra Kamen-škare-papir uz korištenje genetskog algoritma. Rješenja su predstavljena nizom cijelih brojeva. Odluke koje strategija donosi bazirane su na posljednja tri poteza protivnika. Ovakav pristup pokazao se prilično jednostavnim i efikasnim. Genetski algoritam postigao je bolje rezultate od sistematske strategije (determinističkog računalnog algoritma), kao i od ljudskog protivnika.

Postoje i zahtjevniji radovi, poput [6] i [7] gdje se genetski algoritam primijenio za pronalaženje strategija u igrama Dame i Backgammon. Korišteno je nekoliko genotipova: niz logičkih vrijednosti, niz realnih brojeva i aritmetičko stablo. Definirani su specifični genetski operatori za ovaj problem i pomno su odabrani parametri pokretanja kako bi se mogla dobiti dobra rješenja. Dodatno su se koristili i algoritmi pretraživanja prostora stanja. U oba rada zaključeno je da aritmetička stabla daju najbolje rezultate.

Osim standardnih društvenih igara, traženje strategije genetskim algoritmima može se primijeniti na borbene računalne igre, kao što je napravljeno u radu [8]. Riječ je o igri M.U.G.E.N. (Slika 4.1). Cilj je stvoriti borca koji će na osnovu zadnjih nekoliko udaraca protivnika zaključiti koji je najpovoljniji idući potez. Borci (strategije) su potom testirani protiv dvadeset sedam ljudskih protivnika. Rezultati su pokazali da GA daje bolje rezultate od „ručno“ kodiranih algoritama.

Slika 4.1 Igra M.U.G.E.N.

4.2 Idejno rješenje

Imamo definiran problem (pronalaženje strategije), pronašli smo inspiraciju u postojećim rješenjima i imamo alat kojim možemo ostvariti rješenje (evolucijski algoritmi i ECF). Preostalo je glavno pitanje: na koji način iskoristiti alat za rješavanje problema?

Kako bismo odgovorili na ovo pitanje, potrebno je definirati sljedeće:

- Način prikaza rješenja (genotip u ECF-u)
- Funkciju dobrote (način na koji se ocjenjuje svaka jedinka)
- Ulazne parametre
- Povezivanje vrijednosti rješenja s potezima u igri (dekodiranje)
- Broj generacija
- Veličinu populacije
- Dodatne parametre evolucijskog algoritma

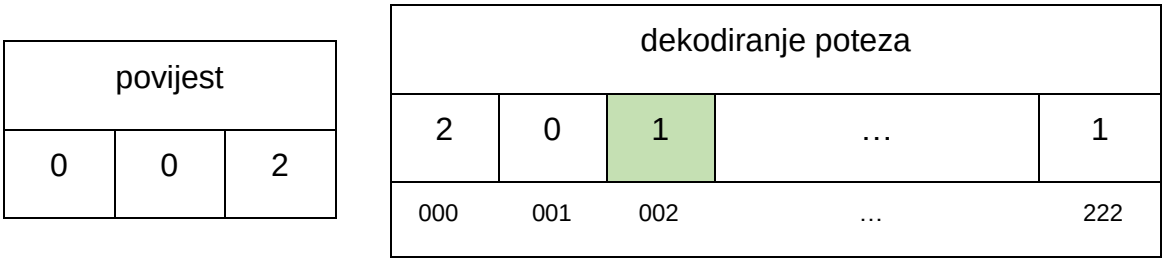
4.2.1 Odabir genotipa

Pri odabiru genotipa treba imati na umu što sve mogu biti ulazni podaci koji će pomoću u učenju strategije, te koji su izlazni podaci koji će na neki način prikazati niz poteza koji čine evoluiranu strategiju.

4.2.1.1 Genetski algoritam – niz realnih brojeva

Prva ideja inspirirana je radom [4]. Izlazne podatke možemo prikazati kao jedan niz realnih brojeva. Ulazni podaci su povijest zadnjih n poteza (npr. 3) protivničkog igrača. Izlazni podaci su potezi koji čine strategiju a izračunavaju se na osnovu povijesti.

Uzmimo n=3 i igru „Kamen-škare-papir“ koja ima 3 moguća poteza. U ovom slučaju postoji 3^3 vrijednosti koje je potrebno dekodirati. Svaki potez dobiva svoju vrijednost, na primjer „kamen“ = 0, „škare“ = 1, „papir“ = 2. Ako je povijest „002“, idući potez čitamo iz trećeg polja u bloku za dekodiranje:



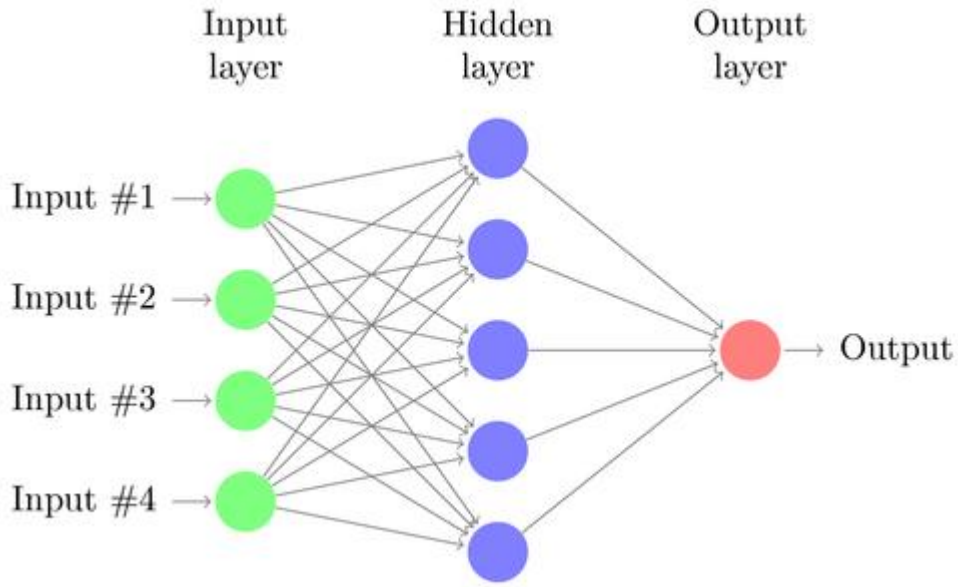
Opisani model ne bi bilo teško izvesti u ECF-u. Potrebno je samo odabrati genotip FloatingPoint kao niz realnih brojeva. Problem je ako želimo mijenjati ulazne parametre ili dodati nešto novo. Tada moramo mijenjati cijeli model, što povlači mijenjanje programskog kôda i ulazne datoteke (ako zbog parametara mijenjamo i veličinu polja).

4.2.1.2 Neuronska mreža

Postoje druga rješenja gdje je jednostavnije promijeniti ili dodati ulazne parametre. Jedno od njih su neuronske mreže.

Bilo koji ulazni podatak koji želimo da utječe na strategiju predstavljat će jedan ulazni neuron. Izlazni neuron je samo jedan – idući potez. Dakle, imamo

mrežu koja na temelju trenutnih vrijednosti definiranih ulaznih parametara kao izlaz daje idući potez strategije.

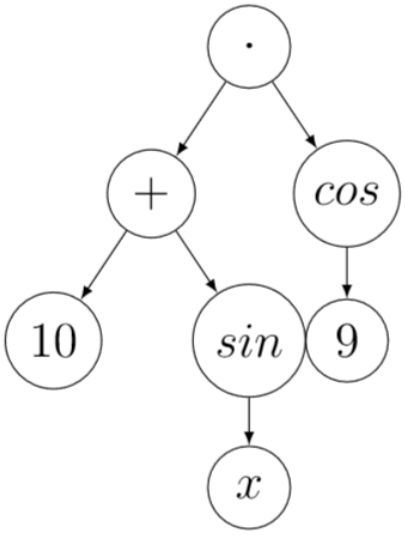


Slika 4.2 Neuronska mreža

U ECF-u postoji genotip neuronske mreže čije se težine uče korištenjem genetskih algoritama [5]. Koristeći ovaj genotip, dodavanje ulaznih parametara provodi se jednostavnije – mijenjanjem strukture mreže u ulaznoj datoteci i eventualno manjim izmjenama programskog kôda.

4.2.1.3 Genetsko programiranje – aritmetičko stablo

Druga opcija s jednostavnijim promjenama ulaznih parametara je korištenje aritmetičkog stabla. Ovakav genotip rješenje prikazuje u obliku aritmetičke funkcije. U varijable funkcije uvrštavaju se ulazni podaci i na taj način dobivamo izlaz – idući potez.



*Slika 4.3 Primjer funkcije (10+sin(x)) * cos(9)*

Dodavanje ulaznih parametara obavlja se u ulaznoj datoteci, uz minimalne izmjene programskog kôda (povezivanje varijabli s vrijednostima ulaza). Prikaz aritmetičkim stablom je prilično intuitivan. Pod uvjetom da znamo kako se dekodiraju različiti potezi, ne bi bilo teško „ručno“ napisati funkciju koja bi predstavljala pojedinu strategiju.

Iz navedenih razloga, u ovom radu koristit ćemo aritmetičko stablo (genotip *Tree* u ECF-u). Problem bi se mogao riješiti i bilo kojim od ostalih navedenih rješenja.

4.2.2 Ocjenjivanje rješenja

Svakoj jedinki (rješenju) trebamo pridijeliti ocjenu dobrote. Jedinke s boljim ocjenama će se propagirati u iduće generacije i na taj način ćemo dobiti rješenje kojim smo zadovoljni.

Kako bismo ocijenili koliko je trenutna strategija dobra, potrebne su nam unaprijed poznate strategije. Naša strategija (jedinka x) odigrat će igru sa svakom od tih strategija (s_i), a suma dobivenih bodova bit će ocjena dobrote te strategije (4.3.1). Kroz generacije tu sumu ćemo pokušati maksimizirati. Ovaj način ocjenjivanja će maksimizirati sumu bodova igrača, bez obzira na protivnika. Ako ne želimo skupiti što više bodova u igri nego biti što bolji u odnosu na protivnika, kao ocjenu dobrote možemo staviti razliku između osvojenih bodova (4.3.2).

$$fit(x) = \sum_{i=1}^{n} bodovi(s_i) \tag{4.3.1}$$

$$fit(x) = \sum_{i=1}^{n} bodovi(s_i) - bodoviProtivnik(s_i) \tag{4.3.2}$$

4.2.3 Ulazni parametri

Postoje razne kombinacije ulaznih parametara koje možemo koristiti pri određivanju strategije. Ovo je dio gdje treba biti kreativan i pronaći što ekspresivnije parametre.

Postoje osnovni parametri koji se nameću pri sastavljanju strategije. To su:

- zadnji potez igrača,
- zadnji potez protivnika i
- broj osvojenih bodova.

Ako želimo da igračima bude poznat broj poteza u igri, može se dodati i parametar koji opisuje koliko je poteza ostalo do kraja.

Osim tih parametara, u obzir možemo uzeti još mnogo toga:

- postotak pobjeda (ukupno ili u zadnjih nekoliko poteza),
- postotak povlačenja određenog poteza kod protivnika,
- je li protivnik u zadnjih n poteza povukao isti potez,
- reagira li protivnik uvijek isto nakon određenog poteza,
- …

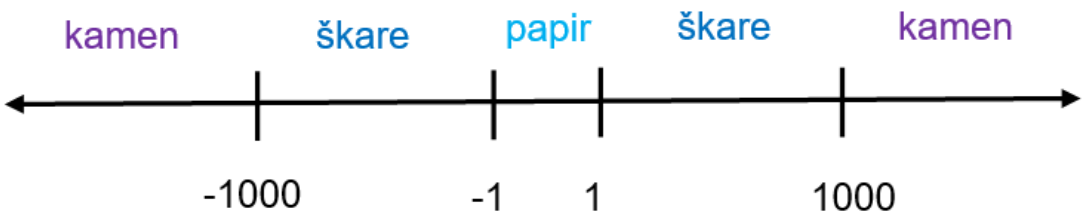
Nisu svi parametri jednako relevantni za pojedinu igru. Neki od njih neće činiti nikakvu razliku u rezultatima. Više o provjeri utjecaja parametara na krajnji rezultat nalazi se u 6. poglavlju.

4.2.4 Prikaz poteza i dekodiranje izlazne vrijednosti aritmetičkog stabla

Budući da radimo s aritmetičkim stablom, poteze u igri moramo moći prikazati u smislenom obliku za aritmetičke funkcije. Svakom potezu možemo pridijeliti neku proizvoljnu cjelobrojnu vrijednost. Nakon isprobavanja nekoliko kombinacija, odlučila sam u Iteriranoj zatvorenikovoj dilemi potezu „izdaja“ pridijeliti broj 1, a „suradnji“ broj 2. u igri Kamen-škare-papir potez „kamen“ predstavlja broj 1, „škare“ 2 a „papir“ 3.

Iduće je pitanje kako interpretirati izlaz aritmetičkog stabla. Nakon što pridijelimo vrijednosti varijablama stabla, dobit ćemo neki realni broj. Potrebno je odrediti intervale za svaki potez. Za igru Iterirana zatvorenikova dilema imamo samo dva poteza koje je potrebno dekodirati. Kao granicu sam uzela |1|, što je prikazano na Slika 4.4. U igri Kamen-škare-papir tri su moguća poteza. Isprobavši razne kombinacije dekodiranja zaustavila sam se na onoj prikazanoj na Slika 4.5.

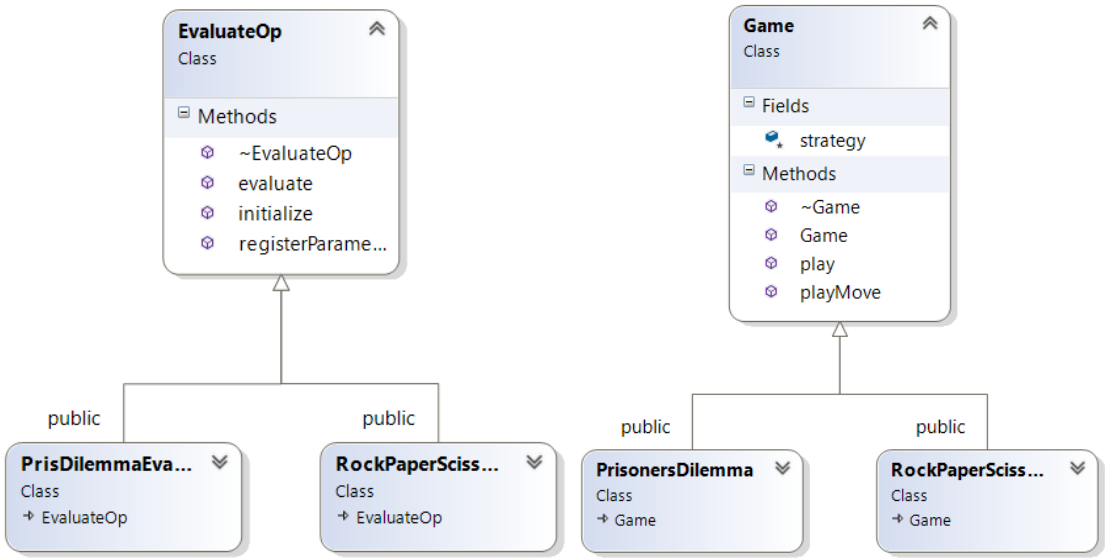
Slika 4.4 Dekodiranje poteza Iterirane zatvorenikove dileme



Slika 4.5 Dekodiranje poteza igre Kamen-škare-papir

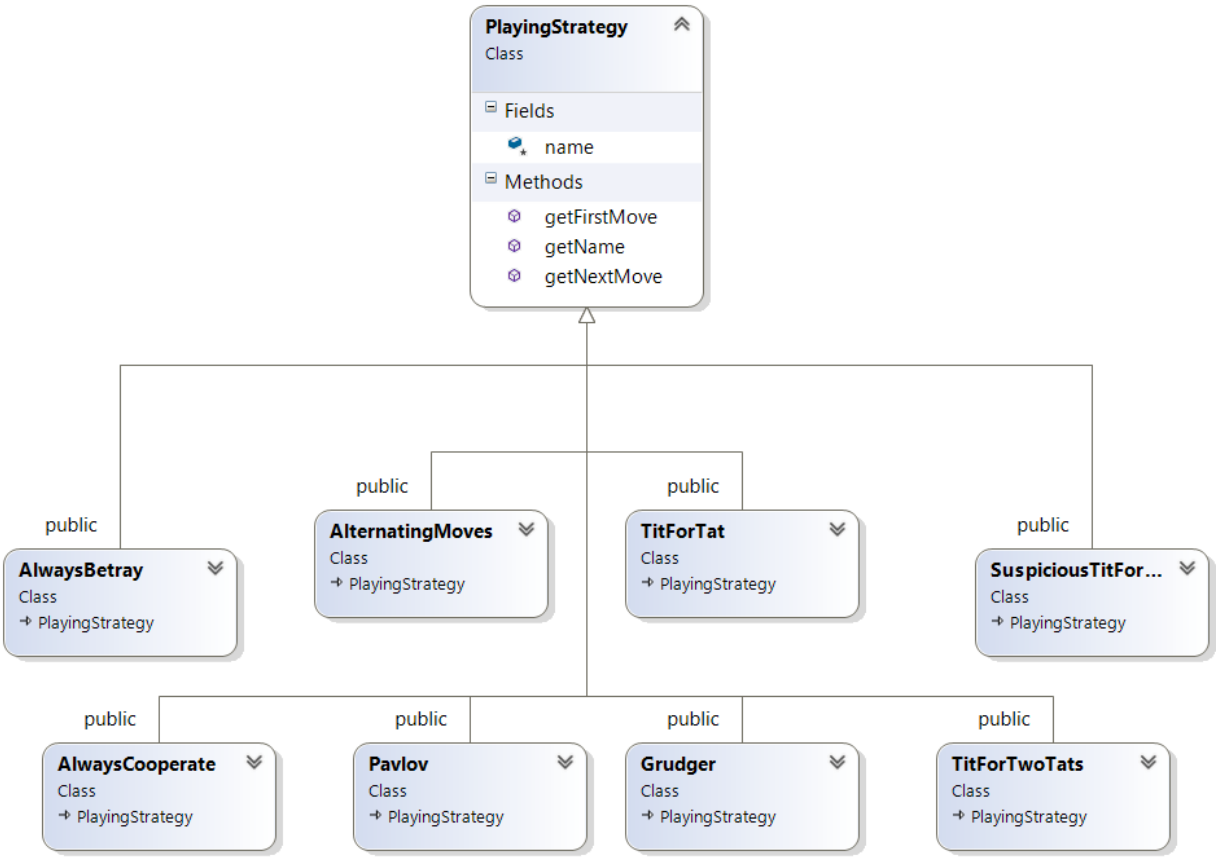
5. Programska implementacija

Problem je programski implementiran tako da se može lako nadograđivati i testirati. Postoji apstraktan razred Game koji treba naslijediti kako bi se dodala nova igra. Trenutno su napravljene igre „Zatvorenikova dilema“ i „Kamen-škare-papir“. Za svaku od igara potrebno je napraviti i evaluacijski operator u kojem se dodjeljuje ocjena dobrote svakoj strategiji. Mogao bi se napraviti i univerzalni evaluator za sve igre, no zbog preglednosti pri testiranju različitih igara sam se odlučila za ovo rješenje. Kako bi se bilo koji operator mogao koristiti u algoritmima ECF-a, treba naslijediti razred EvaluateOp.



Dijagram 5.1 Dijagram razreda za evaluacijske operatore i simulaciju igre

Implementirane su i neke od prethodno opisanih strategija. Koriste se za učenje i ispitivanje evoluiranih strategija. Kako bi se dodalo novu strategiju, potrebno je naslijediti apstraktan razred PlayingStrategy.



Dijagram 5.2 Dijagram razreda za različite strategije

Koristeći opisanu strukturu, pokretanje jedne igre od 100 rundi protiv zadane strategije izgledat će ovako:

```
Game *g = new PrisonersDilemma(new AlwaysCooperate());
g -> play(100, tree, true);
```

6. Rezultati

6.1 Učenje i ispitivanje na jednoj strategiji

Počnimo s jednostavnim pokaznim primjerom na igri Iterirana zatvorenikova dilema. Ideja je provesti učenje samo protiv jedne strategije i zatim provjeriti kako se ponaša dobiveno rješenje protiv te iste strategije. Koristimo osnovni skup ulaznih podataka navedenih u Tablica 6.1, te parametre ECF-a definirane u Tablica 6.2. Uz varijable ulaznih podataka u rješenju mogu se koristiti i konstante između 0 i 1. Uzeli smo malu dubinu stabla kako bi se rješenje moglo preglednije prikazati. Osim standardnih aritmetičkih operatora (+, -, * i /) dodali smo i operatore grananja IFPOS i IFLTE. Funkcija IFPOS prima tri argumenta. Provjerava vrijednost prvog argumenta: ako je pozitivan poprima vrijednost drugog, inače poprima vrijednost trećeg argumenta. IFLTE prima četiri argumenta. Uspoređuje prvi i drugi: ako je prvi veći ili jednak drugome poprima vrijednost trećeg argumenta, u suprotnom će poprimiti vrijednost četvrtog.

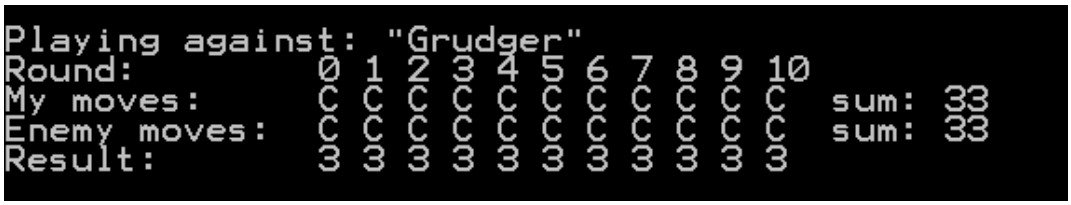
Tablica 6.1 ulazni podaci

x	Moj zadnji potez
y	Zadnji potez protivnika
z	Osvojeni bodovi

Tablica 6.2 Parametri ECF-a

Minimalna dubina stabla	1
Maksimalna dubina stabla	3
Skup funkcija	+, -, *, /, ifpos, iflte
Skup terminala	x, y, z, 0, 1
Veličina populacije	40
Broj generacija	100
Vjerojatnost mutacije	0.3

Uzmimo strategiju „Grudger“ detaljno opisanu u 2. poglavlju. Budući da ona snažno kažnjava „izdaju“, strategija koja donosi najviše bodova je uvijek surađivati. I zaista, uz vrlo malo iteracija i osnovne parametre, uvijek ćemo dobiti očekivan scenarij prikazan na Slika 6.1. Kratica C označava potez „suradnja“ a T „izdaja“.

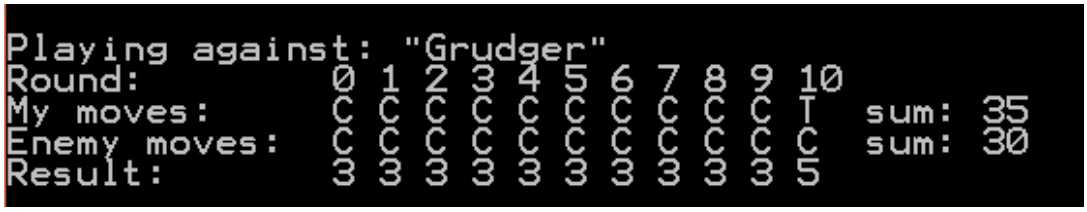


Slika 6.1 Izvođenje protiv iste strategije uz osnovne parametre

Primjer konkretne aritmetičke funkcije koja predstavlja dobivenu strategiju je sljedeći:

$$f(x,y,z) = x*(0.783715*y)+ifpos(y,y,z)$$

Na ovome primjeru može se lijepo vidjeti i što se dogodi ako je duljina igre poznata. Dodavši još jedan parametar (n) koji označava koliko je poteza ostalo do kraja dobivamo scenarij prikazan na Slika 6.2. Strategija je naučila da za „izdaju“ u zadnjem potezu više ne snosi posljedice i tako je maksimizirala broj bodova.



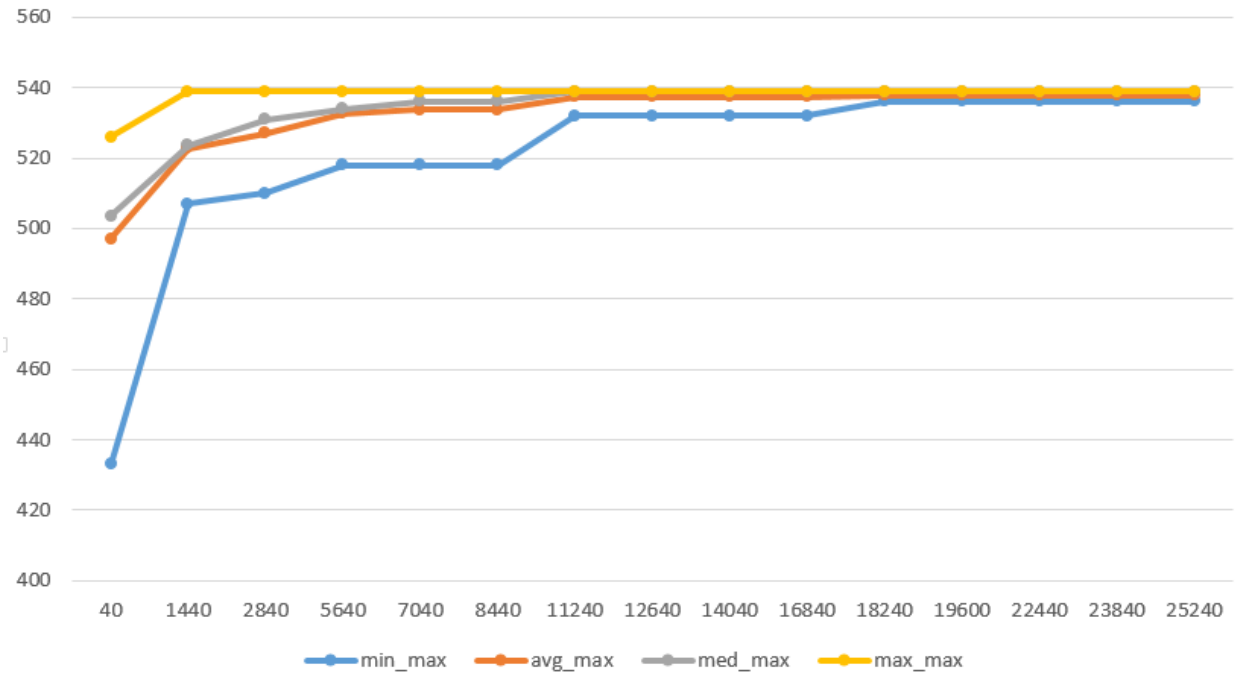
Slika 6.2 Izvođenje protiv iste strategije uz poznati broj poteza

Sada naša funkcija izgleda ovako:

$$f(x,y,z,n) = ((z/y) - (n-x))$$

6.2 Broj generacija

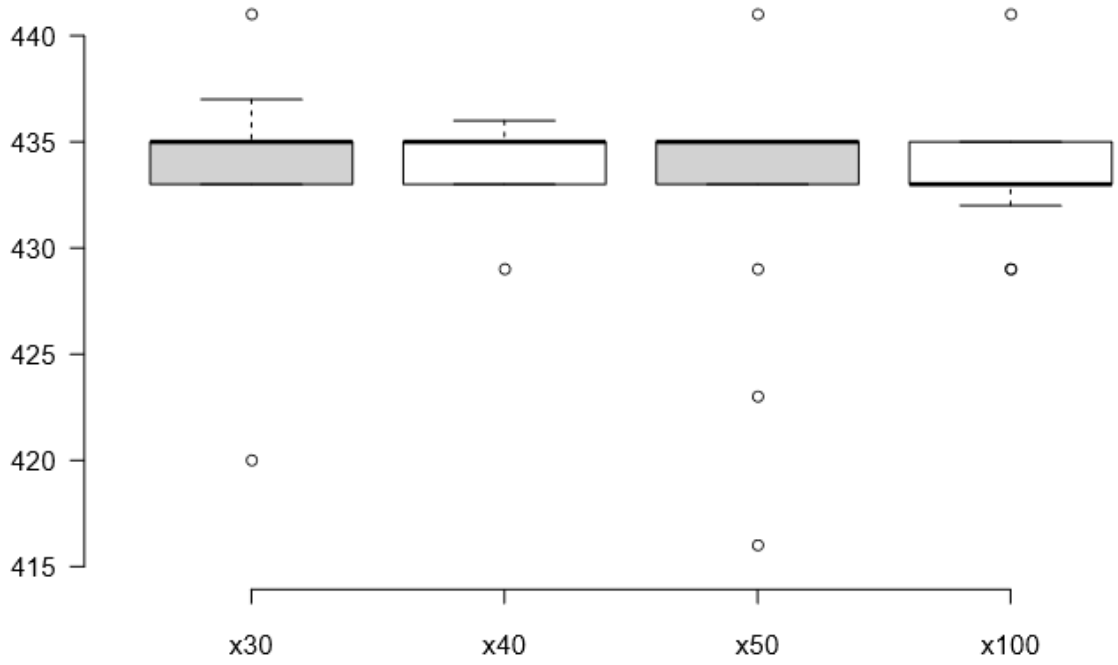
Kad smo se uvjerili da program radi kako bi trebao na učenju jedne strategije, možemo proširiti učenje na više strategija. Učenje ćemo pokrenuti koristeći svih osam implementiranih strategija (vidi 6.5). Potrebno je fiksirati neke važnije parametre ECF-a kako bi svako pokretanje bilo što efikasnije. Prvi takav parametar jest broj generacija. U potrazi za optimalnim rješenjem mogli bismo pustiti da se program izvodi iznimno dugo. No, to neće biti potrebno jer nakon nekog vremena dobrota rješenja više se neće značajno mijenjati. Uzastopnim pokretanjem programa u 10 ponavljanja za igru Iterirana zatvorenikova dilema (po 25 poteza u svakoj igri) došli smo do rezultata prikazanih na Slika 6.3. Zaključujemo da je dovoljno izvođenje zaustaviti nakon otprilike 12000 evaluacija (za još bolje rezultate mogli bismo uzeti oko 19000 evaluacija).



Slika 6.3 Konvergencija maksimuma kroz generacije

6.3 Veličina populacije

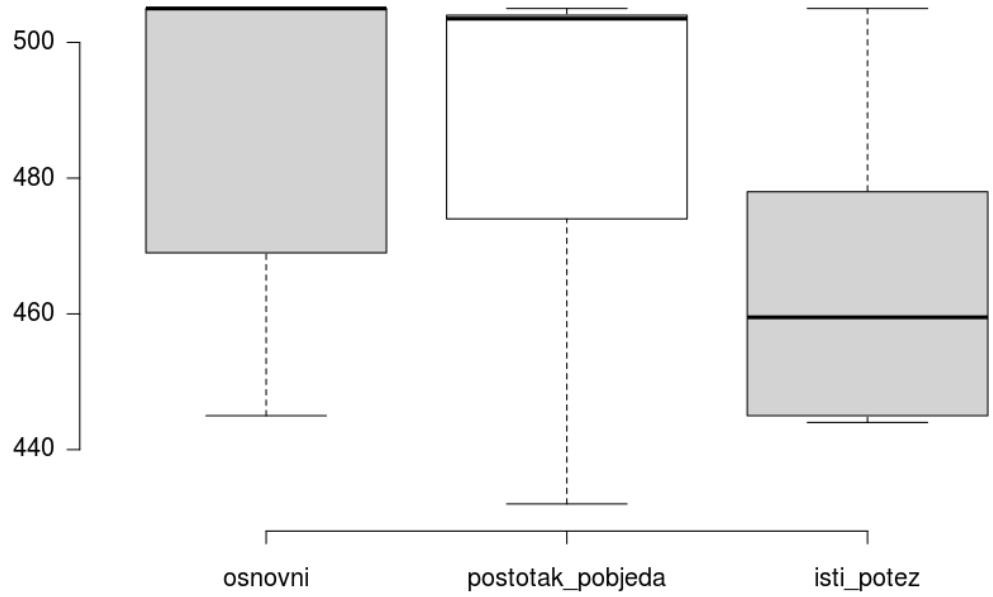
Idući parametar koji možemo optimirati je veličina populacije. Uzastopnim pokretanjem uz različitu veličinu populacije dobiveni su rezultati prikazani na Dijagram 6.1. Možemo zaključiti da su rezultati vrlo slični bez obzira na vrijednost ovog parametra. Uzimamo proizvoljnu vrijednost, recimo 40 jedinki u populaciji.



Dijagram 6.1 Boxplot dijagram rezultata uz različite veličine populacije

6.4 Provjera utjecaja ulaznih parametara

Osim osnovnih parametara navedenih u poglavlju 4.4, postoje mnogi drugi parametri koji mogu utjecati na konačno rješenje. Isprobajmo učenje strategije uz dva dodatna parametra: ukupni postotak pobjeda (broj između 0 i 1) i je li protivnik u zadnja 3 poteza povukao isti potez (zastavica koja poprima vrijednost 0 ili 1). Izvođenjem programa samo s osnovnim parametrima i potom uz dodan po jedan parametar dobili smo rješenje prikazano na Dijagram 6.2. Iz rezultata možemo zaključiti da dodani parametri negativno utječu na krajnje rješenje. Čini se da za igru Iterirana zatvorenikova dilema vrijedi „manje je više“ i algoritam će bolje raditi koristeći samo osnovne parametre.



Dijagram 6.2 Boxplot dijagram utjecaja novih parametara

6.5 Učenje na grupi strategija

6.5.1 Iterirana zatvorenikova dilema

Strategije za učenje podijeljene su u tri grupe prema zajedničkim svojstvima kooperativnosti (Tablica 6.3). Ideja je uzeti svaku grupu zasebno kao grupu strategija za učenje. Dobivenu strategiju možemo primijeniti na igre sa svim preostalim strategijama i usporediti koliko je uspješna protiv određenih grupa. Parametri korišteni pri izvođenju navedeni su u Tablica 6.4.

Tablica 6.3 Strategije podijeljene u grupe

Grupa 1	Grupa 2	Grupa 3
• Always Cooperate • Tit for Tat • Tit for Two Tats	• Alternating Moves • Grudger • Pavlov	• Suspicious Tit for Tat • Always Betray

Tablica 6.4 Parametri ECF-a korišteni pri učenju na grupama strategija

Minimalna dubina stabla	1
Maksimalna dubina stabla	5
Skup funkcija	+, -, *, /, ifpos, iflte
Skup terminala	x, y, z, 0, 1
Veličina populacije	40
Broj generacija	300
Vjerojatnost mutacije	0.3

U Tablica 6.5 možemo vidjeti rezultate učenja na prvoj grupi strategija. To su vrlo kooperativne strategije gdje se do najpovoljnijeg ishoda dolazi uvijek surađivajući. Rezultati unutar iste skupine su dobri, protiv druge skupine su osrednji, a protiv zadnje strategije naša strategija je očekivano podbacila, jer je stalna suradnja ovdje najgori izbor.

Tablica 6.5 Rezultati pri učenju s prvom grupom

UČENJE: GRUPA 1	Prosjek	Max	Min	Protivnikov prosjek
Always Cooperate	44,14	55	33	16,29
Tit for Tat	27,71	33	15	26,29
Tit for Two Tats	34,57	45	32	26,71
Alternating Moves	22,71	32	18	31,29
Grudger	21,71	33	6	40,29
Pavlov	29,71	35	8	29,00
Suspicious Tit for Tat	25,43	30	14	29,00
Always Betray	2,86	10	1	43,57
Suma:	208,86			242,43

Druga skupina za učenje sakupila je najviše bodova (Tablica 6.6) a treća (Tablica 6.7) najmanje. Druga skupina uspješno je na strategiji Grudger naučila važnost surađivanja, a zbog strategije Pavlov uspjela je „pametnije“ reagirati na poteze protivnika, odnosno povlačiti one poteze koji su se pokazali uspješnima.

Tablica 6.6 Rezultati pri učenju s drugom grupom

UČENJE: GRUPA 2	Prosjek	Max	Min	Protivnikov prosjek
Always Cooperate	43,00	55	33	18,00
Tit for Tat	24,86	33	15	22,00
Tit for Two Tats	29,57	35	19	25,29
Alternating Moves	31,57	35	28	15,14
Grudger	25,29	33	15	23,14
Pavlov	32,43	35	23	19,29
Suspicious Tit for Tat	14,00	26	11	14,00
Always Betray	9,86	11	5	15,57
Suma:	210,57			152,43

Za lošije rezultate treće skupine vjerojatno je kriva „narav“ dviju strategija: iako obje započinju s „izdajom“, protiv strategije Suspicious Tit for Tat ipak se više isplati surađivati. To je razlog i zašto su strategije iz prve skupine ovdje sakupile više bodova od protivnika.

Tablica 6.7 Rezultati pri učenju s trećom grupom

UČENJE: GRUPA 3	Prosjek	Max	Min	Protivnikov prosjek
Always Cooperate	34,14	35	33	31,29
Tit for Tat	32,71	33	32	32,71
Tit for Two Tats	33,86	35	33	31,71
Alternating Moves	19,43	20	18	40,86

Grudger	17,00	33	5	42,71
Pavlov	21,00	33	5	40,29
Suspicious Tit for Tat	21,86	30	11	24,71
Always Betray	3,14	11	0	42,43
Suma:	183,14			286,71

Učenje sa svim strategijama polučilo je zadovoljavajući rezultat (Tablica 6.8). Ipak je skupljeno manje bodova nego učenjem s drugom grupom: te su se strategije pokazale najneutralnijima i mogu dobro aproksimirati ponašanje drugih strategija.

Tablica 6.8 Rezultati pri učenju sa svim strategijama

UČENJE: SVE STRATEGIJE	Prosjek	Max	Min	Protivnikov prosjek
Always Cooperate	42,43	55	33	18,86
Tit for Tat	27,86	33	15	26,43
Tit for Two Tats	23,00	33	19	17,29
Alternating Moves	25,29	35	18	31,00
Grudger	27,86	33	15	26,43
Pavlov	34,14	35	33	17,00
Suspicious Tit for Tat	19,14	30	11	21,29
Always Betray	3,14	11	0	42,43
Suma:	202,86			200,71

6.5.2 Kamen-škare-papir

Naposljetku, pogledajmo kako se isti pristup ponaša na drugoj igri. Uzmimo implementaciju igre Kamen-škare-papir, nekoliko osnovnih strategija i pokrenimo učenje na cijeloj grupi strategija. Parametri ECF-a ostali su isti (Tablica 6.4). Iz

rezultata prikazanih u Tablica 6.9 možemo zaključiti da naučena strategija prilično dobro pokriva svaku od ovih osnovnih strategija.

Tablica 6.9 Rezultati za Kamen-škare-papir

UČENJE: SVE STRATEGIJE	Prosjek	Max	Min	Protivnikov prosjek
Tit for Tat	13,71	25	10	10,29
Scissors Only	19,14	33	10	7,57
Paper Only	18,14	33	0	12,57
Rock Only	17,00	33	0	9,29

7. Zaključak

Nakon provedenog istraživanja, napravljene programske implementacije i testiranja možemo zaključiti da je moguće pronaći strategije igranja ponovljenih igara pomoću evolucijskih algoritama. Budući da nije sasvim jednostavno „ručno“ odrediti koje su dobre strategije za pojedine igre, evolucijski algoritmi koji se oslanjaju na nasumičnost ovdje su se pokazali kao koristan alat. Strategije dobivene ovim postupkom mogu konkurirati dobrim strategijama do kojih se došlo na službenim turnirima. Dobivena rješenja mogla bi se i poboljšati novim parametrima i detaljnijim ispitivanjem.

Prikazane su samo jednostavne igre, no model bi se mogao primijeniti i na složenije: šah, dama, poker, Backgammon… U tom slučaju bi bilo potrebno proširiti model, napraviti neke preinake i osigurati veću procesorsku moć.

Literatura

[1] Davis, W., *Iterated Prisoner's Dilemma Online Game and Simulation*, 22.3.2007, http://www.iterated-prisoners-dilemma.net, 26.5.2017

[2] Chen, J., Lu, S., Vekhter, D. *Game Theory – Axelrod's Tournament* https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/axelrod.html, 26.5.2017

[3] Jakobović, D. *ECF - Evolutionary Computation Framework,* 30.10.2016, http://ecf.zemris.fer.hr/, 26.5.2017

[4] Ali, F., Nakao, Z. Chen, Y., *Playing the Rock-Paper-Scissors Game with a Genetic Algorithm* https://pdfs.semanticscholar.org/bca1/3d0856130714531f2d1a83e3a997b4ab7ebd.pdf, 26.5.2017

[5] Bejuk, B., Pavlek, D., Škorjanc, I., Fulir, J., Jaklinović, K., Ulaga, L., Burda, M., Lovrenčić, R., *ECF - NeuralNetwork,* https://ecf-neural.herokuapp.com/, 26.5.2017

[6] Benbassat, A., Sipper, M., *Evolving Lose-Checkers Players using Genetic Programming,* http://ieeexplore.ieee.org/document/5593376/, 7.6.2017

[7] Azaria, Y., Sipper, M., *GP-Gammon: Genetically Programming Backgammon Players,* Genetic Programming and Evolvable Machines, Vol. 6, Broj 3 (2005.), str. 283-300

[8] Martinez-Arellano, G., Cant, R., Woods, D. *Creating AI Characters for Fighting Games using Genetic Programming DOI 10.1109/TCIAIG.2016.2642158, IEEE Transactions on Computational Intelligence and AI in Games*

Pronalaženje strategije u igrama s više igrača uz pomoć evolucijskih algoritama

Sažetak

U radu je objašnjena teorija ponovljenih igara s naglaskom na Iteriranu zatvorenikovu dilemu. Pojašnjene su strategije u toj igri, kako se ponašaju kooperativne i nekooperativne strategije. Ukratko je opisana glavna ideja evolucijskih algoritama. Objašnjeni su svi koraci potrebni za stvaranje programske implementacije: odabir genotipa, definicija funkcije dobrote, biranje ulaznih parametara, prikazivanje poteza u igri, traženje broja generacija te optimalne veličine populacije. Opisan je način na koji je organizirana programska implementacija. Naposljetku su prikazani dobiveni rezultati i zaključci.

Ključne riječi: ponovljene igre, igre s više igrača, optimizacija strategije, evolucijski algoritmi, genetsko programiranje

Strategy evolution in multiplayer games using evolutionary algorithms

Abstract

This paper describes the repeated games theory with emphasis on the Iterated Prisoners Dilemma game. This game's strategies are illustrated: how are the cooperative and non-cooperative strategies behaving. The main idea behind the evolutionary algorithms is briefly described. The paper goes through all the steps needed for creating the software implementation: choosing the genotype, defining the fitness function, choosing the input parameters, representing the game moves, finding the necessary number of generations and the optimal population size. It describes the organization of the implementation components. Finally, the results and conclusions are presented.

Keywords: repeated games, multi-player games, strategy optimization, evolutionary algorithms, genetic programming