Гимназија

Младеновац, Краља Александра Обреновића 25

Матурски рад из програмирања

СИМУЛАЦИЈА ЕВОЛУЦИЈЕ НА РАЧУНАРУ

Ментор:

Предраг Матовић, проф.

Ученик:

Лазар Стошић, IV-6

Младеновац, јун 2022. год.

С А Д Р Ж А Ј

Страна

1. УВОД

Еволуција у природи је једна јако комплексна појава, коју чини мноштво различитих фактора и која траје дуги временски период, због чега ју је тешко у потпуности проучити. Међутим, само проучавање еволуције има своје примене у мноштву различитих области, од медицине и фармације, па све до вештачке интелигенције и машинског учења.

Еволуциони алгоритам у рачунарској интелигенцији означава врсту оптимизационог алгоритма заснованог на моделу популације. У њему се користе принципи из биолошке еволуције, као што су генетика, размношавање, мутација, рекомбинација и селекција да би се што тачније и ефикасније решио неки проблем. "Еволуција" у овом смислу се догађа кроз поновно примењивање горе наведених операција. Заснован примарно на Дарвиновој теорији еволуције, у овом алгоритму решења проблема се добијају преко скупа индивидуалних организама, тј. популације. Свака јединка има свој приступ решавању проблема и самим тим даје могућност да се ефикасније размотри више различитих приступа.

Иако овакав алгоритам има велики потенцијал за налажење различитих решења, он често уме да буде и захтеван за извршавање због своје комплексности. Ово се, међутим, може побољшати бољом оптимизацијом, конкретно тзв. *fitness* апроксимацијом. Иако није увек најефикасније решење, овај алгоритам несумњиво има своје примене у решавању захтевнијих проблема. Комплексност алгоритма није увек директно везана за комплексност проблема, па тако и наизглед прост алгоритам може решити и врло захтевне проблеме.

Циљ овог рада је да кроз рачунарску симулацију еволуције **„ EvoSim’’** покаже начин рада оваквог алгоритма. Ова симулација је упрошћена верзија знатно напреднијих и ефикаснијих симулација, чиме се показују најосновнији принципи рада самог алгоритма.

2. ЕВОЛУЦИОНИ АЛГОРИТАМ

2.1. Основе рада еволуционог алгоритма

Еволуциони алгоритам, конкретно једна од његових грана, звана **генетски алгоритам,** се заснива на следећа два корака:

Први корак – Ствара се примарна популација са насумично генерисаним генетским особинама.

Други корак – Понављање следећих корака до краја извршавања алгоритма:

1. Проверавање сваке јединке да ли испуњава услове за преживљавање.
2. Бирање најспособнијих јединки за размножавање (родитељи).
3. Стварање нових јединки насумичним укрштањем гена родитеља, као и случајном мутацијом.
4. Замењивање јединки које нису преживеле у прошлој генерацији са новим, способнијим јединкама.

2.2. Врсте еволуционих алгоритама

Генетски алгоритам – најпопуларнији облик еволуционог алгоритма. Решава проблеме преко генетског кода, тј. стринга бинарних цифара које одређују његове генетске особине. На њега се примењују рекомбинација и мутација.

Генетско програмирање – овде су решења дата у облику компјутерских програма и њихов квалитет ("fitness") одређује њихова способност да реше неки рачунарски проблем. Постови више подврста генетског програмирања, као што су картезијанско генетско програмирање, линеарно генетско програмирање и граматичка еволуција.

Еволуционо програмирање – слично генетском програмирању, али је овде структура програма фиксна, а параметри унутар њега могу да еволуирају.

Еволуциона стратегија – ради са вектором реалних бројева који представљају решења и обично користи само-прилагодљиву мутацију.

Неуроеволуција – слично генетском програмирању, али се генотипом представља вештачка неуронска мрежа, описивањем веза између чворова (неурона) и њихових тежина/вредности. Кодирање гена може бити директно и индиректно.

3. СИМУЛАТОР ЕВОЛУЦИЈЕ „EVOSIM"

3.1. Шта је „EvoSim"?

„EvoSim" је кумпјутерски симулатор еволуције. Он ради по принципу **генетског алгоритма** и **неуроеволуције** да представи развиће простих, виртуелних организама кроз коначан број генерација. У њему је преко графичког интерфејса представљено кретање свих јединки у ограниченом простору(мапи) и који су задати услови за њихово преживљавање (доћи до одређеног дела мапе).

Организми раде по принципу неуронских мрежа, које су запрво графови веза између **сензора("чула"), акција("радње')** и **неурона**, чворова који регулишу везе између чула и акција. У зависности од генетског кода, чула и радње ће бити различито повезане. Неуронска мрежа ефективно врши улогу мозга организма. Сви организми имају иста урођена чула и радње, али у зависности од њихових гена, само неки од њих ће бити активни.

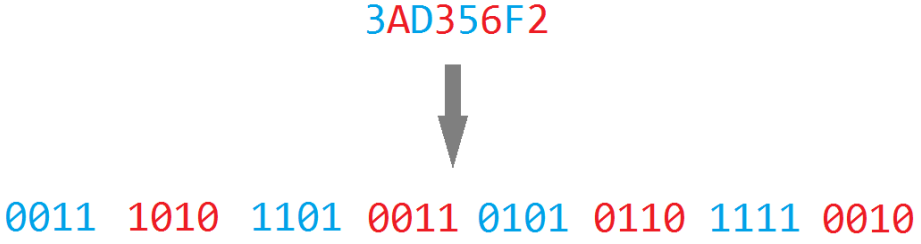
Организми извршавају акције у корацима, где у једном кораку сваки организам извши један сет акција. Сваког корака се узимају вредности активних чула и даље обрађују ради извршавања потребних радњи.

Након краја сваке генерације, тј. након извршеног одређеног броја корака, евалуира се позиција на мапи. Они који испуњавају задати услов ће насумично стварати парове, и сваки пар ће дати једног потомка, који има случајно расподељене гене од оба родитеља.

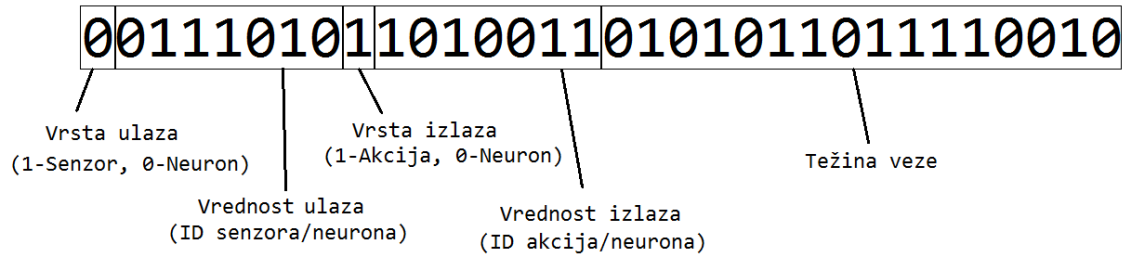
3.2. Генетски код

Сваки организам има свој генетски код, тј. **генотип**, представљен кроз низ хексадецималних цифара. Низ од 8 хексадецималних цифара чини један **ген**. Сваки ген представља једну везу између чворова у неуронској мрежи. Тај осмоцифрени број се претвара у бинарну основу и тиме се добија запис од 32 бита. Он се даље претвара у везе између чворова, по правилима на слици 2.

Вредности ИД-јева су обично мале, отприлике до 20, тако да се узима остатак дељења датог броја са укупним бројем сензора/акција/неурона. Тежина везе је број који се претвара у децималну вредност у опсегу -8,0 до 8,0 (вредност је узета произвољно).

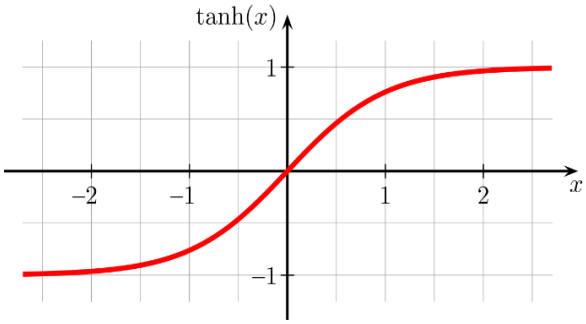


Слика 1.1. – Пример конверзије из хексадецималног записа у бинарни.

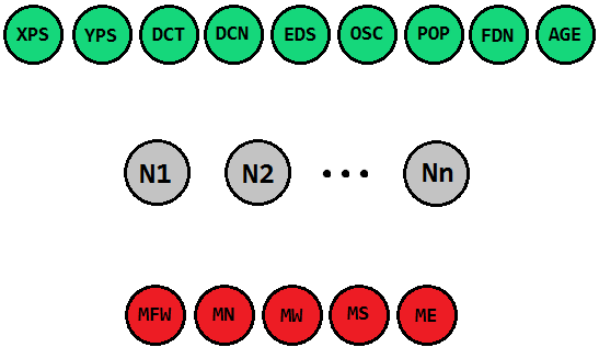


Слика 1.2. – Пример генетског кода и његово читање.

Свако од активних чула исказује интензитет којим је активирано децималним бројем од 0,0 до 1,0. Та вредност се даље прослеђује акцији или неурону и свака од њих се множи са тежином везе. Потом сваки од активних неурона рачуна збир свих вредности које улазе у њега и пропушта их кроз **хиперболичну тангентну функцију**. Ова функција ће дат вредности искључиво у опсегу од -1,0 до 1,0.



Слика 1.3. - Графички приказ хиперболичне тангентне функције



Слика 1.4. – Шематски приказ свих чворова у неуронској мрежи

3.3. Чула и акције

Способности ових организама су ограничене на следећа чула и акције:

Чула:

XPS – X позиција организма у арени
YPS – Y позиција организма у арени
DCT – Раздаљина од центра арене
DCN – Раздаљина од (0,0) позиције
EDS – Раздаљина од најближе ивице
OSC – Осцилатор (синусна функција која за вредност узима старост организма плус насумичну вредност од 0 до 1)
POP – Густина популације у *n* радијусу

FDN – Густина хране у *n* радијусу (неискоришћено)
AGE – Старост организма у односу на трајање генерације

Акције:

MFW – Организам се помера у истом правцу као и у прошлом кораку (уколико је слободно)
MN – Организам се помера за једно поље нагоре (уколико је слободно)
MW – Организам се помера једно поље налево (уколико је слободно)
MS – Организам се помера једно поље надоле (уколико је слободно)
ME – Организам се помера једно поље надесно (уколико је слободно)

Неурони:

Број неурона (*n*) може бити задат пре почетка извршавања програма.

3.4. Параметри

Пре почетка симулације, могу се задати различити параметри који се односе на саму симулацију. Они су:

- Број неурона сваког организма (подразумевана вредност: **4**)
- Број гена сваког организма, што самим тим одређује и дужину генетског кода (подразумевана вредност: **16**)
- Дужина генерације у корацима (подразумевана вредност: **300**)
- Број јединки у генерацији (подразумевана вредност: **500**)
- Да ли је дозвољена мутација? (подразумевана вредност: **тачно**)
- Уколико је дозвољена, шанса да дође до мутације при преносу сваког засебног гена (подразумевана вредност: **0.001 = 0.1%**)
- Да ли ће програм да сам од себе изврши одређен број генерација (подразумевана вредност: **тачно**)
- Број генерација који ће програм извршити сам од себе (подразумевана вредност: **100**)

4. О СТРУКТУРИ ПРОГРАМА

4.1. Основно о структури програма

„EvoSim" је програм писан потпуно у C++ програмском језику, ревизија 2017. Као развојно окружење, коришћен је Visual Studio Community 2019, а програмски код је чуван и контролисан преко GitHub платформе за контролу верзија.

Графички интерфејс је писан у OpenGL-у, једном од најпопуларнијих програмских интерфејса за рад са рачунарском графиком. Конкретно, коришћена је SDL (Simple DirectMedia Layer) библиотека, која је базирана на OpenGL-у.

Код програма је подељен по библиотекама и ово је претежно урађено због разумљивости самог кода. Сам програм користи принципе као што су објектно-оријентисано програмирање, рад са графиком и рад са фајловима.

Програм се покреће из **Source.cpp** фајле, где се налази редефинисана *main* функција, зато што SDL користи своју верзију *main* функције. Програм се извршава преко класе Program, која се налази у Program.h библиотеци. Та класа садржи пет главних метода које се позивају:

Init – метода која иницијализује прозор за исцртавање графичког интерфејса, прослеђују јој се параметри за име прозора, X и Y позицију горњег левог угла прозора, његове димензије у пикселима и да ли се он приказује преко целог екрана. У **Source.cpp** фајлу је позвана на следећи начин:

```
prog = new Program();
prog->Init("Program", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, 800, 400, false);
```

HandleEvents – проверава да ли је дошло до неких промена/догађаја од предходног догађаја, конкретно да ли је прозор програма затворен и да ли је дошло до упита на тастатури.

Update – извршава промене у параметрима, позива се у сваком кораку. Регулише понашање организама по корацима и обрађује објекте потребне за исцртавање.

Render – исцртава графички приказ симулације на екрану сваког корака.

Clean – затвара прозор симулације и брише га из меморије.

Методе **Init, HandleEvents** и **Update** се позивају у сваком кораку.

```
while (prog->Running()) {
	frameStart = SDL_GetTicks();

	prog->HandleEvents();
	prog->Update();
	prog->Render();
}
```

У **Source.cpp** фајлу постоју и део кода који регулише брзину извршавања програма. Може се ограничити на четири брзине: 0.5x (15 корака по секунди), 1x (30 корака по секунди), 2x (60 корака по секунди) и MAX који не ограничава број корака по секунди, већ их извршава по највећој брзини коју дозвољавју спецификације рачунара.

4.2. Библиотеке

Од већ постојећих библиотека, искоришћена је једино **SDL.h** библиотека за графички интерфејс. Корисничке библиотеке које служе искључиво за овај програм су:

- **Program.h** – већ описана библиотека. Садржи класу **Program**, која регулише извршавање целе симулације.
- **Global.h –** садржи неке основне промењиве и функције које остатак програма користи, као што је структура **Coord**, која представља XY координату на мапи.
- **Param.h** – садржи већ наведене параметре који одређују извршавање симулације.
- **Randomize.h** – садржи функцију која враћа bool вредност тачно у зависности од свог параметра, тј. реалног броја од 0 до 1, где тај број представља вероватноћу да функција врати вредност true. Такође садржи функцију која генерише насумичан генетски код.
- **Genome.h** – садржи структуру **Gene** и дефинише вектор тих структура као **Genome.** Такође садржи функцију која претвара генетски код у **Gene** структуру.
- **Neuron.h** – садржи енумерацију врста сензора и акција**.**
- **Organism.h** – садржи класу **Organism,** чије инстанце запрво представљају појединачне организме у симулацији.
- **Survival.h** – садржи две функције. Једну која за задату координату проверава да ли она испуњава услов за пролаз, тј. да ли се налази у првих ***n*** поља са леве или десне стране мапе. Друга за задат вектор објеката враћа само оне који испуњавају задати услов.
- **InitGen.h** – садржи функцију која од задатог вектора преживелих јединки креира нове организме насумичним спајањем парова организама из претходне генерације (родитеља), и враћа их у форми вектора. Уколико је дозвиљена мутација, при рекомбинацији сваког гена постоји врло мала шанса да се један његов део насумично промени.

5. РЕЗУЛТАТИ СИМУЛАЦИЈЕ

При покретању програма, видимо да је понашање јединки насумично. Већина њих заврши близу ивица мапе, али је то само из разлога зато што се генерално крећу у једном правцу и ограничени су ивицом. Самим тим, јединке су релативно равномерно распоређене уз све четири ивице. Мањи број њих се задржи на средини или се не креће уопште.

Узмимо да је услов преживљавања да се јединке налазе уз првих 15 поља уз леву и десну ивицу. Ако јединкама доделимо подразумеване параметре (4 неурона, 16 гена, 500 јединки у генерацији и 300 корака по генерацији), овако изгледа стопа преживљавања првих 100 генерација:

БРОЈ ГЕНЕРАЦИЈЕ	СТОПА ПРЕЖИВЉАВАЊА
1	62.4%
2	60.4%
3	63.4%
4	67.8%
5	73.2%
6	73.4%
7	71.2%
8	78%
9	73.6%
10	70.2%

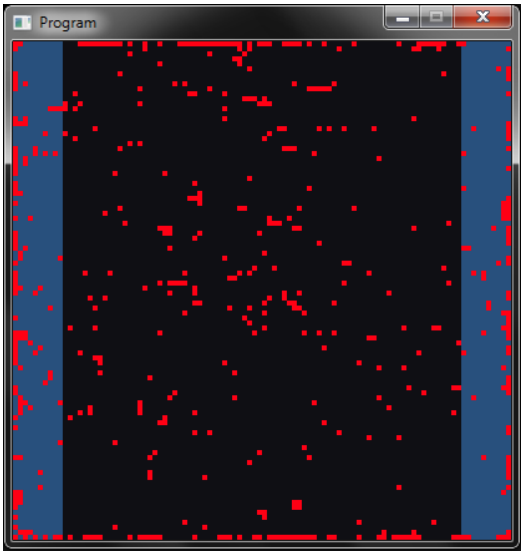
БРОЈ ГЕНЕРАЦИЈЕ	СТОПА ПРЕЖИВЉАВАЊА
10	70.2%
20	80%
30	89.8%
40	89.8%
50	94.4%
60	97.4%
70	96%
80	99.4%
90	98.6%
100	99%

Графикон 1 – Стопа преживљавања кроз генерације

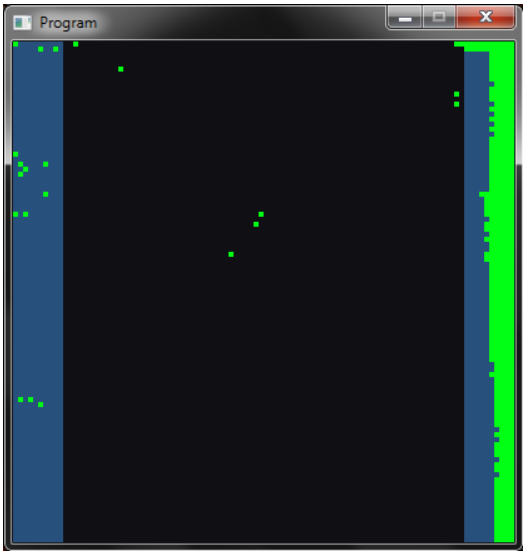
Како можемо видети, стопа преживљавања генерално расте. Јасно се види да постоје моменти када она опада и моменти када расте – нису све јединке условно боље од својих родитеља. Постоји довољно простора да све јединке преживе уколико се само крећу ка једној од две "сигурне зоне". Пратећи јединке кроз генерације, јасно се види да обично преовладава једно одређено понашање, као што је "крећи се искључиво на лево/десно", међутим, понекад се јављају и групе које се одвајају од већине и одлазе у супротну страну.

Гени који их доводе до "сигурне зоне" се преносе кроз потомство, а гени који им то не омогућавају полако нестају, јер се губе кроз јединке које се нису размножиле. Самим тим, биодиверзитет врсте после одређеног броја генерација је врло мали.

У случају да променимо услове преживљавања, или да поставимо неку препреку, организмима би било врло тешко, ако не и немогуће да се адапттирају. Једино што би им то омогућило је активација неких скрајнутих гена, или врло вероватније, мутација.



Слика 2.1 – Изглед симулације, у току 1. генерације.



Слика 2.2 – Изглед симулације, у току 100. генерације.

6. МАНЕ ”EVOSIM” СИМУЛАЦИЈЕ

6.1. Мане са биолошког гледишта

Ако поредимо дату симулацију са биолошком еволуцином, лако се може видети да је ово знатно упрошћена верзија праве еволуције. Постоји енорман број параметара који одређују како се права еволуција одвија, а ова симулација је само једна врста апрокцимације праве еволуције. Генерације се у природи не смењују једнаком брзином, време не пролази у корацима, неки гени могу бити доминантни а неки рецесивни, постоји већа интеракција између јединки, услови за преживљавање се мењају и ако се не адаптирају на њих довољно, врсте изумиру.

Не постоји симулација довољно напредна да обради све те параметре, а и спрам оних које већ постоје и покушавају бар да се што више приближе томе, ова симулација је врло упрошћена.

6.2. Мане са рачунарског гледишта

Овај програм је направљен да ради сам релативно малим бројем јединки и симулира врло просте услове њихових живота. У случају да се имплементирају напредније функције Коришћен је C++ програмски језик зато што важи за један од најефикаснијих виших програмских језика. У ’’Evosim’’ симулацији је просечно време извршавања једне генерације од 300 корака између 1,6 и 2,3 секунде, у зависности од броја неурона и дужине генетског кода. Међутим, при неким већим прорачунима, који би захтевали хиљаде или милионе организама и генерација, овај приступ постаје скоро неупотребив.

7. ЗАКЉУЧАК

Симулације ове врсте се могу користити за много различитих потреба, од симулирања популације микроорганизама, што може помоћи лекарима да боље разумеју њихове животне процесе, економије, где се користи за предвиђање цена акција на берзи, па све до грана инжињерства као што су машинство, роботика, али и интердисциплинарне области као што је *data science*.

Ова симулација служи као једна врста увода у начине за решавање оваквих проблема. Преко ње можемо видети како да створимо организме са одређеним особинама и да регулишемо њихово понашање.

Хтео бих овим путем да се захвалим свом ментору Предрагу Матовићу, који је са мном прошао кроз моју идеју и дао ми смернице о томе у ком смеру да је даље развијем, као и како да је адаптирам што боље за овај рад.

ПРИЛОГ

Прилог 1

https://github.com/cyancat529/EvoSim - линк ка github страници пројекта.

ЛИТЕРАТУРА

1. https://en.wikipedia.org/wiki/Evolutionary_algorithm
2. https://towardsdatascience.com/introduction-to-evolutionary-algorithms-a8594b484ac
3. https://github.com/davidrmiller/biosim4

Датум предаје: _____

Комисија:

Председник _____

Испитивач _____

Члан _____

Коментар:

Датум одбране: _____ Оцена_____ (___)