UNIVERSITATEA ”ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

XYZ

descriere XYZ

propusă de

Denise-Mihaela Goldan

Sesiunea: Iulie, 2018

Coordonator științific

Conf. dr. Adrian Iftene

UNIVERSITATEA ”ALEXANDRU IOAN CUZA” IAȘI

FACULTATEA DE INFORMATICĂ

XYZ

Denise-Mihaela Goldan

Sesiunea: Iulie, 2018

Coordonator științific

Conf. dr. Adrian Iftene

Declarație privind originalitatea și respectarea drepturilor de autor

Prin prezenta declar că Lucrarea de licenţă cu titlul "XYZ" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţă precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, XY Iulie 2018 Denise-Mihaela Goldan

Declarație de consințământ

Prin prezența declar că sunt de acord ca Lucrarea de licență cu titlul "XYZ", codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iaşi, XY Iulie 2018 Denise-Mihaela Goldan

Cuprins

Problema iterată a prizonierului

1 Enunţul clasic al dilemei prizonierului

Dilema prizonierului[1] reprezintă o problemă tratată în teoria jocurilor.

A fost formulată în anul 1950 de către Merrill Flood și Melvin Dresher, angajați ai companiei RAND Corporation[2]. Denumirea (dilema prizonierului) este meritul lui Albert W. Tucker, de la Universitatea Princeton, care a formalizat jocul și a introdus noțiunea de **răsplată** (engl. payoff).

Enunțul clasic al problemei este prezentat în paragraful următor:

> *Doi suspecți sunt arestați de către poliție. Polițiștii nu au suficiente dovezi pentru a condamna suspecții, așa că îi duc în camere separate și le propun amândurora aceeași ofertă. Dacă unul dintre suspecți depune mărturie pentru urmărirea penală împotriva celuilalt suspect și celălalt tăinuiește faptele, cel care a trădat este eliberat și cel care a tăinuit primește o pedeapsă de 10 ani de închisoare. Dacă niciunul nu mărturiseşte, ambii ajung în pușcarie pentru jumătate de an. Dacă se trădează reciproc, fiecare primește o pedeapsă de 5 ani. Suspecții au de ales între a trăda și a tăinui faptele.*

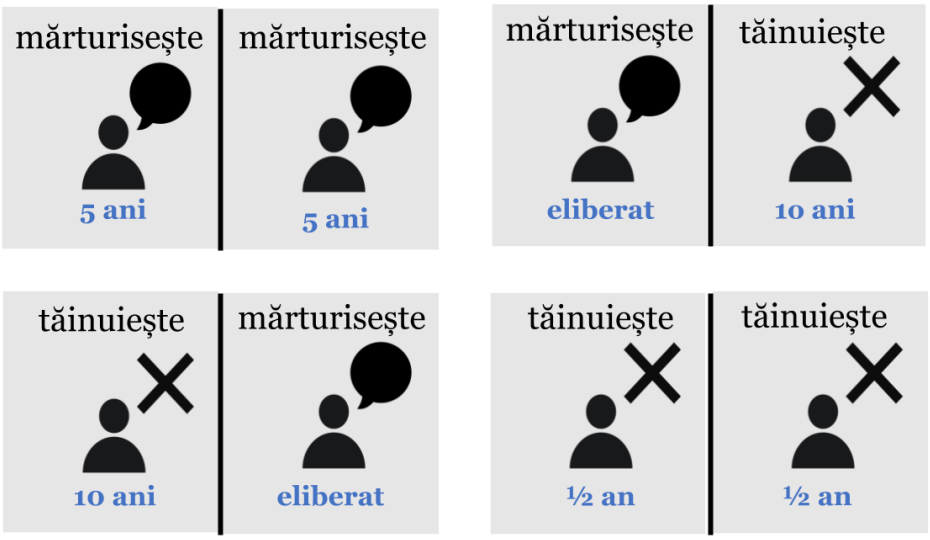

Figura 1: *Prezentare a celor patru posibilități descrise în scenariul anterior.*

Observaţii: Considerăm că atunci când un suspect mărturisește, mărturia să îl incrimineză pe celălalt suspect. Tăinuirea faptelor reprezintă decizia suspectului de a nu spune nimic anchetatorilor.

Putem formaliza aceast paragraf prin următorul tabel al recompenselor, unde cei doi suspecți sunt numiți **A** și **B**[3]. Se prezintă rezultatele obținute pentru fiecare combinație dintre tăinuire și marturisire:

Acţiune aleasă		**Rezultat obţinut**	
A	**B**	**A**	**B**
tăinuiește	tăinuiește	**Reward**	**Reward**
tăinuiește	mărturiseşte	**Sucker's payoff**	**Temptation**
mărturiseşte	tăinuiește	**Temptation**	**Sucker's payoff**
mărturiseşte	mărturiseşte	**Punishment**	**Punishment**

Tabelul 1: *Adaptare după matricea recompenselor pentru dilema prizonierului*

Termenii care apar în tabel sunt următorii:

- **Temptation**: recompensa obținută de jucatorul ce mărturisește atunci când celalalt tăinuiește faptele
- **Reward**: recompensa pentru când cei doi suspecți, A și B, aleg să tăinuiască
- **Punishment**: pedeapsa obținută de fiecare dintre cei doi suspecți atunci când se trădează reciproc
- **Sucker's payoff**: pedeapsa pentru cel care a tăinuit atunci când celălalt l-a trădat

Pentru a face trecerea de la pedeapsa cu închisoarea la ideea de joc, considerăm că cei 4 termeni reprezintă scoruri de valoare pozitivă și că scopul suspecților este să obțină un număr cât mai mare de puncte.

Între acești termeni ce reprezintă recompensele, se respectă următorul lanț de inegalități:

Temptation >Reward >Punishment >Sucker's payoff

1.1 Strategii pentru dilema prizonierului

Dacă suspectăm că celălalt reținut va tăinui faptele, suntem mai avantajați dacă mărturisim (vom fi "răsplătiți" cu **Temptation**, despre care știm că are o valoare mai mare decât **Sucker's payoff**- scorul pe care îl va obține celălalt suspect), decât dacă tainuim (caz în care toți primesc **Reward**; **Temptation** >**Reward**).

Dacă suspectăm că celălalt va mărturisi împotriva noastră, suntem mai avantajați dacă mărturisim și noi (vom primi amândoi **Punishment**), decât dacă tainuim (caz în care primim **Sucker's payoff**, însă celalt va primi un scor mai bun, **Temptation**).

În alte cuvinte, indiferent de mișcarea celuilalt jucător, *strategia care ne avantajează câștigul personal, în defavoarea câștigului celuilalt jucător, este mărturisirea împotriva acestuia.*

Dilema este dată de faptul că ambii suspecți ar fi obținut un scor mult mai bun dacă ar fi tăinuit amândoi (protejând pe celălalt participant), decât dacă amândoi ar fi ales să mărturisească (în defavoarea celuilalt suspect). Pe scurt, *cooperarea celor doi aduce cel mai mare avantaj de ambele părți.*

2 Varianta iterată a dilemei prizonierului

Dacă s-ar juca mai multe runde, în care ambii jucători ar alege să se trădeze reciproc la fiecare rundă, scorul pe care l-ar obține ar fi mult mai mic decât dacă ar alege să tainuiasca faptele în fiecare rundă.

În teoria jocurilor, problema iterată a prizonierului este catalogată drept joc cu suma nenulă [1] (engl. non-zero-sum game).

Un meci între doi jucători este reprezentat de un număr de runde, care nu este cunoscut de catre participanți.

În fiecare rundă, cei doi jucatarori aleg independent, în secret, ce mișcare vor face: vor tăinui sau vor mărturisi. La final de rundă, cei doi își expun alegerea. În funcție de ce mișcare au ales cei doi, fiecare este răsplătit cu un anumit câștig, care se adaugă la scorul total individual al jucătorilor.

[1] Numim joc de sumă nenulă jocul în care suma câstigurilor este diferită de zero.

Dacă la o anumită rundă ambii au cooperat, scorul ambilor jucători va crește cu o valoare ce poartă denumirea de **Reward**.

Dacă ambii trădează, vor primi **Punishment payoff**.

Dacă unul cooperează, dar celălalt trădează, cel care a cooperat primește **Sucker's payoff** și celălalt este răsplătit cu **Temptation**.

2.1 Strategii pentru problema iterată a prizonierului

Considerând acest scenariu drept un joc, folosim termenul de **cooperare** (engl. cooperation) pentru a descrie situația când unul dintre suspecți **tăinuiește** faptele. **Mărturisirea** faptelor de către un suspect pentru incriminarea celuilalt suspect va fi numită **trădare** (engl. defection).

Turneele lui Axelrod

Printre studiile pofesorului Robert Axelrod, care predă științe politice și politici publice la Universitatea din Michigan, se află și problema iterată a prizonierului.

Interesul său de a afla o strategie potrivită l-a determinat să organizeze două turnee de tip *fiecare cu fiecare* (engl. round-robin). În aceste turnee, fiecare participant joacă pe rând împotriva tuturor celorlalți[4]. Primul turneu a inclus 14 programe ce furnizează strategii de joc, iar cel de al doilea a avut un număr de 63 de programe. Observațiile sale sunt trecute în lucrarea *The Evolution of Cooperation*, scrisă în 1984[6]. Axelrod a solicitat participanților strategii sub formă unor programe care cunosc istoricul ultimelor trei runde.

Câștigătorul ambelor turnee a fost strategia **Tit-for-Tat**. Această strategie cooperează la prima rundă, apoi, în următoarele runde, utilizează mișcarea făcută de oponent în runda anterioară. În alte formă de idei, cooperează de fiecare dată când celălalt cooperează și trădează când este trădată, dar nu inițiază trădarea.

Strategii analizate

În următoarele rânduri, sunt enumerate câteva strategii analizate:

- **Always cooperate**: Jucătorul cooperează la fiecare rundă a jocului, indiferent de strategia aplicată de celălalt jucător.
- **Always defect**: Jucătorul trădează la fiecare rundă a jocului.
- **Grudger**: Această strategie presupune cooperarea la fiecare rundă, până la prima trădare din partea celuilalt jucător. Așadar, adoptând această strategie, dacă oponentul trădează chiar și o singură dată, următoarele mișcări, până la final de joc, vor fi de trădare.
- **Pavlov**: Se alege cooperarea la prima rundă. Dacă la runda anterioară jucătorul a fost recompensat cu **Temptation**[2] sau **Reward**[3], acesta repetă ultima mișcare. În celălalt caz, alege mișcarea opusă.
- **Random**: Se alege la întâmplare următoarea acțiune.
- **Tit-For-Tat**: Se alege cooperarea la prima rundă. De la runda a doua, jocuatorul ce alege această strategie repetă ultima mișcare a oponentului.
- **Suspicious Tit-For-Tat**: Diferența dintre această strategie și **Tit-For-Tat** este că la prima mișcare se alege trădarea.
- **Tit-For-Two-Tats**: Jucătorul cooperează de fiecare dată, făcând excepție acele cazuri în care jucătorul este trădat de două ori consecutiv.

[2]**Temptation** este recompensa obținută de jucatorul ce trădează atunci când oponentul cooperează.

[3]**Reward** reprezintă recompensa primită de ambii jucători atunci când cooperează.

Algoritm genetic

1 Apariția noțiunii de algoritm genetic

Algorimii genetici[6] au fost introduși de către John Holland în 1960 și dezvoltați, ulterior, alături de colegii de la Universitatea din Michigan, între anii 1960 și 1970. Holland urmărea înțelegerea fenomenului de adaptare întâlnit în natură și implementarea unor mecanisme adaptive care să fie utilizate în practică, în contextul programării. Cartea publicată de acesta în 1975, *Adaptation în Natural and Artificial Systems* (Holland, 1975/1992) prezintă algoritmii genetici drept abstractizări ale evoluției biologice și oferă un cadru teoretic pentru dezvoltarea acestora.

Algoritmii genetici ai lui Holland sunt metode de a trece de la o populație de cromozomi (e.g., șiruri de biți care reprezintă soluții candidat pentu o problemă) la o nouă populație, prin folosirea selecției, alături de operatorii inspirați din genetică: încrucișare, mutație, inversiune. Cea din urmă este rar folosită în practică.

> *Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand.*
>
> John H. Holland

Algoritmii genetici au fost creați în încercarea de a imita procese specifice evoluției naturale, cum ar fi lupta pentru supraviețuire și moștenirea materialului genetic. Putem privi evoluția drept strategia abordată de speciile biologice pentru a căuta soluții cât mai potrivite, adaptate condițiilor schimbătoare, într-un număr foarte mare de posibilități. Această abordare poate fi utilizată în rezolvarea problemelor de optimizare, atunci când metodele clasice exhaustive nu se dovedesc eficiente.

Noțiunea de algoritm genetic nu este definită în mod riguros[5], însă toate metodele ce poartă această denumire au în comun următoarele: populația este formată din cromozomi, selecţia este făcută pe baza rezultatelor funcției de optimizat, încrucișarea a doi *cromozomi părinți* produce doi *cromozomi copii*, mutația se aplică *cromozomilor copii*.

2 Schema generala a unui algoritm genetic

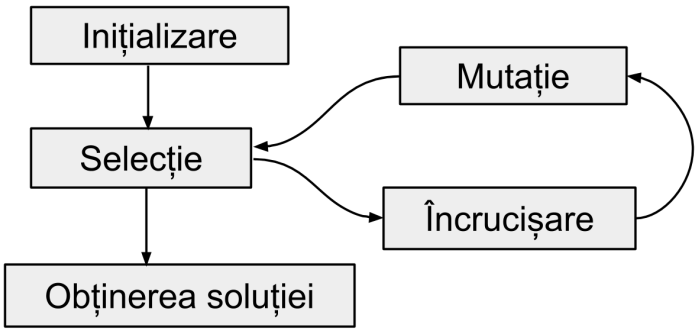

Figura 2: *Schema prezintă componentele de bază ale unui algoritm genetic și legăturile dintre acestea.*

3 Pseudocod

```
inițializează cu valori aleatorii populația inițială
calculează valoarea funcției de optimizat pentru indivizii populației
cât timp nu s-a îndeplinit condiția de oprire
     aplică o metodă de selecție, pentru a crea populația
     aplică operatorul genetic încrucișare, cu o anumită probabilitate
     aplică operatorul genetic mutație, cu o anumită probabilitate
     calculează valoarea funcției de optimizat pentru indivizii populației
```

Condiția de oprire poate fi atingerea unui număr de iterații stabilit inițial. De asemenea, se poate stabili ca algoritmul să se oprească atunci când nu se mai înregistrează îmbunătățiri în ceea ce privește calitatea soluțiilor furnizate.

Soluția returnată de un algoritm genetic reprezintă cel mai bun individ întâlnit în evoluția populației.

4 Terminologie

- Soluțiile candidat sunt adesea codificate în forma unor șiruri de biți și se mai numesc **cromozomi** sau **indivizi** ai populației. Fiecare bit este echivalentul unei gene.
- Genele sunt informațiile stocate de către cromozomi.
- **Populația**, care va fi urmărită în procesul său evolutiv, este alcătuită din mai mulți cromozomi.
- Fiecare **generație** marchează câte o etapă din evoluția populației inițiale.
- Pentru a trece de la o generație la alta, apelăm la noțiunea de **reproducere**. În alcătuirea următoarei generații, se pornește de la populația actuală, pe care o supunem unui proces de **selecție**. Pentru a face analogia cu fenomenul de supraviețuire a celor mai adaptați indivizi, măsurăm cromozomii cu ajutorul unei **funcții de optimizat**. O valoare ridicată a acestei funcții este interpretată ca o bună adaptare la mediu a individului.

- Pentru explorarea spațiului de soluții, indivizii selectați suferă modificări. Sunt supuși **încrucișărilor** și **mutațiilor**.
- Încrucișarea combină genele a doi *cromozomi părinți*, rezultând doi **moștenitori**. Există mai multe variante: cu un punct de tăiere, ales aleator (în care un moștenitor este alcătuit dintr-o porțiune de cromozom de la primul părinte și o porțiune de la al doilea), cu mai multe puncte de tăiere și uniformă (unde fiecare genă este selectată probabilist de la unul din cei 2 *cromozomi părinți*).

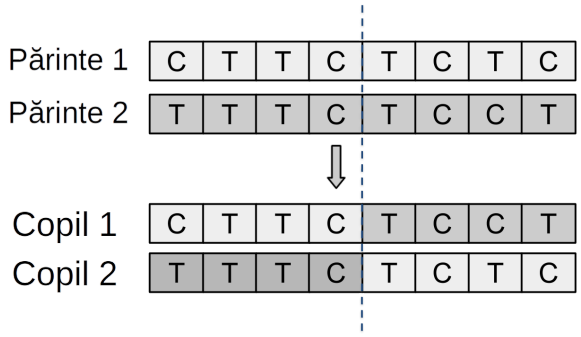

Figura 3: *Exemplu de încrucișare cu un singur punct de tăiere. Punctul de tăiere este marcat prin linia verticală punctată, între elementele de la indexul trei și patru din crmozomi.*

- Mutația alterează gene alese arbitrar dintr-un cromozom. Numărul de gene afectate poate varia.

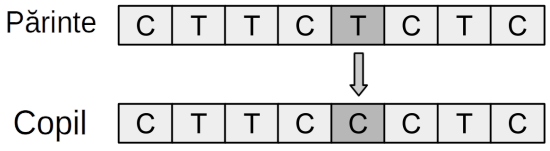

Figura 4: *Mutaţia are loc la poziţia cu indexul cinci din cromozomul părinte.*

Dezvoltarea unei strategii pentru problema iterată a prizonierului folosind un algoritm genetic

1 Tehnologii folosite

Ca și limbaj de programare, am ales să implementez soluția problemei folosind limbajul **Java**, care are o suită performantă de biblioteci.

Pentru ca procesul de obținere a unei noi strategii să fie vizibil prin loguri, am apelat la biblioteca **SLF4J**[11][4].

Pentru partea de vizualizarea a datelor, pentru a vedea cum evoluează strategiile pe parcursul unui turneu cu eliminare, am apelat la biblioteca integrată **JavaFX**[10].

Pentru manipulatea fișierelor json, am folosit biblioteca **json-simple**[12].

2 Reprezentarea soluției

În urma organizării celor două turnee, Axelrod[7] a decis să cerceteze dezvoltarea unei strategii pentru problema iterata a prizonierului, folosindu-se de algoritmii genetici introduși de Holland.

[4] Abrevierea provine de la Simple Logging Facade for Java.

Unul din cei mai importanți pași din acest proces a fost stabilirea unei modalități de a reprezenta o strategie în forma unui cromozom. Descriem concluziile lui Axelrod în rândurile următoare.

Presupunem că fiecare jucător are capacitatea de a memora mișcările ultimei runde sub forma unei perechi: primul element reprezintă mișcarea proprie, iar cel de al doilea element reprezintă mișcarea oponentului. Ne vom folosi de următoarea notație:

C reprezintă **cooperarea cu oponentul**
T reprezintă **trădarea oponentului**

Există patru perechi (sau cazuri) posibile:

Cazul 1: **CC**
Cazul 2: **CT**
Cazul 3: **TC**
Cazul 4: **TT**

Pentru acest scenariu, strategia reprezintă ce mișcare vom alege, dat fiind aflarea mișcării oponentului în ultima rundă.

Strategia **Tit-for-Tat** este reprezentată în felul următor:

dacă **CC** atunci **C**
dacă **CT** atunci **T**
dacă **TC** atunci **C**
dacă **TT** atunci **T**

Dacă impunem ca aceste patru cazuri să respecte ordinea lexicografică, putem codifica strategia drept șirul de caractere **CTCT**. Ca să utilizăm această reprezentare a strategiei:

- Observăm ce a ales oponentul în runda anterioară;

- Formăm perechea compusă din mișcarea noastră, împreună cu cea a oponentului;
- Vedem indexul care corespunde perechii obținute la pasul anterior;
- Alegem mișcarea pe care o găsim la indexul respectiv.

Strategiile lui Axelrod se bazau pe istoricul ultimelor trei runde. Pentru acestea, există 64 [5] de posibile scenarii pentru utlimele trei runde:

Cazul 1: **CC CC CC**
Cazul 2: **CC CC CT**
Cazul 3: **CC CC TC**
...
Cazul 62: **TT TT CT**
Cazul 63: **TT TT TC**
Cazul 64: **TT TT TT**

Ca și în ipoteza în care jucătorii memorează doar istoricul ultimei runde, putem reprezenta aceste cazuri într-un șir de caractere de lungime 64. Vom folosi un șir de caractere de lungime 71, pentru a reține și ce mișcări ar trebui făcute în primele runde, când încă nu există un istoric care să cuprindă ultimele trei runde. Cele 7 poziții de la începutul șirului de caractere au următoarele semnificații:

1. La **poziția numărul 1** se găsește mișcarea aleasă pentru prima rundă a jocului;
2. **Poziția numărul 2**: mișcarea pentru cea de a doua rundă, dacă la rundă anterioară oponentul a cooperat (în istoricul oponentului se găsește doar mișcarea notată cu **C**);
3. **Poziția numărul 3**: mișcarea pe care o vom face la cea de a două rundă, dacă la rundă anterioară oponentul a trădat (în istoricul oponentului se găsește doar mișcarea notată cu **T**);

[5] Numărul de șiruri unice de caractere pe care le putem genera folosind doar caracterele **C** și **T** este 64, sau 2^6.

4. **Poziția numărul 4**: mișcarea pe care o vom face la cea de a treia rundă, dacă pentru primele două runde, istoricul oponentului este **CC**;

5. **Poziția numărul 5**: mișcarea pentru cea de a treia rundă, dacă pentru primele două runde, istoricul oponentului este **CT**;

6. **Poziția numărul 6**: mișcarea pentru cea de a treia rundă, dacă pentru primele două runde, istoricul oponentului este **TC**;

7. **Poziția numărul 7**: mișcarea pentru cea de a treia rundă, dacă la primele două runde oponentul a avut mișcările **TT**.

3 Dimensiunea spațiului de căutare

Având 71 de poziții pe care le putem ocupa cu cele două caractere, putem genera 2^{71} șiruri de caractere distincte. Acest număr reprezintă numărul tuturor strategiilor pe care îl putem avea, în contextul în care cunoaștem istoricul ultimelor trei runde ale jocului.

Spațiul de căutare este, în concluzie, mult prea mare pentru a caută exhaustiv cea mai bună strategie.

4 Funcția de optimizat

Axelrod a alcătuit un set de opt strategii cu care să concureze fiecare strategie a algoritmului genetic, în vederea calculării unei funcții de optimizat. Acest set de strategii nu include strategia **Tit-for-Tat**. Valoarea funcției de optimizat este dată de media scorurilor obținute în urma meciurilor jucate cu fiecare dintre ele opt strategii.

În implementare, am considerat că **Temptation** payoff are valoarea 5, **Reward** are valoarea 3, **Punishment** are valoarea 1 iar **Sucker's payoff** are valoarea 0[15].

5 Parametrii algoritmului genetic

Prin utilizarea unui algoritm genetic, pot obține o copie a celui mai bun individ din toate generațiile care au participat la antrenare. În alte cuvinte, acest cromozom conține strategia care a obținut cea mai bună valoare a funcției de optimizat.

Pentru crearea acestui individ, este nevoie de ajustarea mai multor parametri și opțiuni, dintre care menționez: rata mutației, rata încrucisării, tipul selecției populației [6]. Cromozomul are capacitatea de a-și formula următoarea mișcare bazându-se pe istoricul ultimelor trei runde. Din acest motiv, este important ca un meci să fie format din mai multe runde. Așadar, numărul de runde reprezintă și acesta un parametru pentru antrenarea cromozomilor.

6 Configurația unui algoritm genetic

Numim **configurație a algorimului genetic** un cumul de perechi cheie-valoare, unde cheia reprezintă numele unui parametru, iar valoarea reprezintă valoarea pe care alegem să o atribuim parametrului.

Parametrii despre care discutăm sunt: dimensiunea populației de antrenare, numărul de generații, rata mutației, rata încrucișării, configurația populației de antrenament, numărul de runde jucate în fiecare meci[7].

7 Populația de antrenament și de test

Avem nevoie de două fișiere de configurare pentru stabilirea populației de antrenament și de test. În cadrul proiectului, se regăsesc sub denumirea de

[6] Populația diferă ușor de la generație la generație, selectandu-se doar anumiți indivizi și în anumite proporții

[7] Pentru a calcula valoarea funcției de cost pentru un cromozom, acesta va participa într-un **turneu clasic** alături de o populație de antrenament, definită în fișierul **training.config.json**, alcătuită din diverse strategii standard. Cromozomul va juca un meci format dintr-un număr de runde cu fiecare din membrii populației de antrenament. Valoarea funcției de cost va fi dată de scorul obținut la final de turneu și este calculată drept suma scoruruilor obținute de cromozom la fiecare meci.

testing.config.json și **training.config.json**. Fișierele pot fi ușor manipulate prin clase specializate în citirea și scrierea lor. În fiecare din aceste două fișiere se vor găsi numele unor strategii standard, alături de numărul de indivizi din acel tip de strategie.

Exemplul de mai jos conține definiția unei populații formate din câte un jucător din umatoarele strategii: **Always Cooperate**, **Always Defect**, **Grudger**, **Pavlov**, **Tit-For-Tat**, **Suspicious Tit-For-Tat**, **Tit-For-Two-Tats**.

Exemplu:

```
{
        "Always Cooperate": 1,
        "Always Defect": 1,
        "Grudger": 1,
        "Pavlov": 1,
        "Tit-For-Tat": 1,
        "Suspicious Tit-For-Tat": 1,
        "Tit-For-Two-Tats": 1
}
```

Toate strategiile dezvoltate prin intermediul algoritmului genetic sunt salvate într-un director de resurse, în fișiere în format json, alături de datele relevante pentru dezvoltarea cromozomilor:

* strategia rezultată
* o medie a scorului din turneul clasic
* numărul de runde jucate în fiecare meci
* configurația populației de antrenament (sub forma exemplificată mai sus)
* numărul de generații
* dimensiunea populației antrenate
* probabilitatea de încrucișare
* probabilitatea de mutație

Odată stabilite valorile parametrilor, se poate trece la inițializarea în mod aleator a cromozomilor și antrenarea lor. În final, vrem să obținem într-un fișier strategia cea mai bine adaptată la condițiile de antrenament. Procesul de inițializare și antrenare se repetă de un număr de ori pentru aceeasi configuratie a algoritmului genetic. Experimental, se poate observa că **restartarea algoritmului** poate duce la reținerea unor rezultate mai bune decât dacă am lăsa algoritmul să ruleze o singură dată.

Timp de un număr de generații, cromozomii sunt supuși unor transformări:

- se aplică un proces de selecție
- se aplică operatorul încrucișării
- se aplică mutația

Procesul de selecție pe care am decis să îl implementez este selecția de tip ruletă (engl. roulette wheel). Această selecție este un algoritm stocastic, în care indivizii sunt distribuiți pe niște segmente contigue. Lungimea segmentelor este direct proportională cu cât de bine este adaptat la mediu individul. Fiecare segment determină un interval. Se generează pe rând câte un număr aleator și este ales intervalul (care corespunde unui unic individ-practic se alege individul) la care aparține numărul. Individul selectat se adaugă în mulțimea ce va înlocui, la finalul selecției, generația actuală. Procesul seamănă cu învârtirea ruletei, unde fiecare felie a ruletei are mărimea direct proportională cu gradul de adecvare[9].

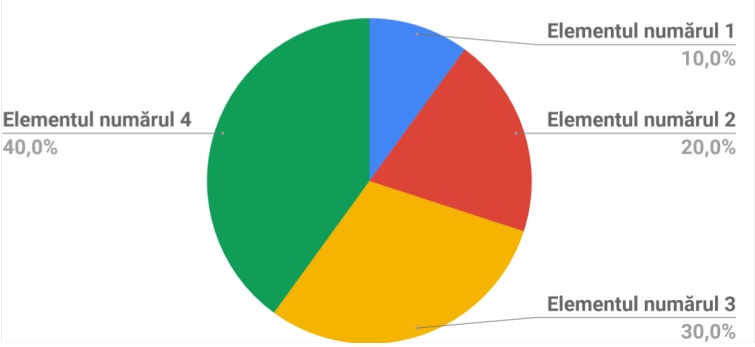

Figura 5: *Probabilități de selecție ale elementelor.*

Figura 5 reprezinta un exemplu de ruletă divizată în patru sectoare de arii proporționale cu probabilitățile de selecție[14]. Elementul numărul 1 are 10% șanse de a fi selectat. Elementul numărul 2 are șanse de 20%, numărul 3 de 30% iar numărul 4 de 40%.

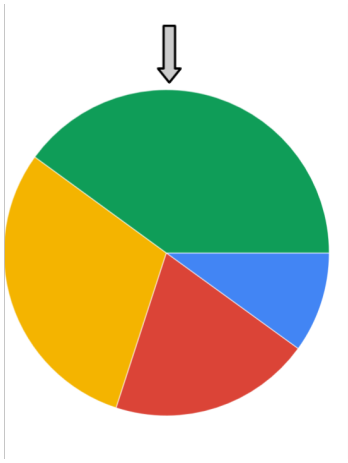

Figura 6: *Rezultatul rotirii ruletei.*

Rezultatul din figura 6 poate fi interpretat drept selectarea individului care corespunde categoriei etichetate cu verde (se alege elementul cu numarul 4, care a avut o probabilitate de selectie de 40%).

Turnee eliminatorii între strategii

Ne interesează să găsim *cea mai bună strategie*. Pentru a compara între ele mai multe strategii, trebuie să gândim *un mediu în care acestea să concureze*.

Până la a găsi cea mai bună strategie, trebuie, mai întâi, să vedem cum anume poate configurația unui algoritm genetic să influențeze calitatea soluției. De asemenea, vrem să vedem cum anume se comportă într-un mediu de test strategia propusă de algoritm.

Pentru a îndeplini aceste cerințe, am ales să supun cromozomii la un anumit tip de turneu, ce poartă denumirea de **turneu cu eliminare**.

1 Termeni întâlniţi

O **rundă** este dată de alegerea, în mod secret, a mișcării următoare și actualizarea scorului în funcție de ce a pus și oponentul.

Un **meci** este jucat de către doi jucători. Este alcătuit dintr-un număr de runde. În fiecare rundă, fiecare jucător alege, în mod secret, ce mișcare va face. La final de rundă, scorul jucătorilor este actualizat cu o valoare dată de mișcarea făcută de fiecare, în funcție de ce a ales și oponentul să facă.

2 Cum este modelat un turneu cu eliminare

Un turneu cu eliminare pornește de la o populație de strategii în care, la fiecare iterație, fiecare individ joacă câte un meci cu ceilalți indivizi. Pe parcursul meciurilor, câștigurile individuale se însumează într-un scor total. După ce se joacă toate combinările de doi jucători (se ajunge la finalul iterației), se elimină un procent din cei mai slabi jucători. Pentru a mai reduce din numărul parametrilor ale căror valori pot varia, am stabilit că procentul să fie de 25%. În caz de egalitate a scorurilor între doi jucători, se elimină la întâmplare unul din cei doi. Se completează locurile eliberate cu stategii care au obținut printre cele mai bune scoruri. Se resetează scorul total al indivizilor și se repetă acești pași până când în turneu a rămas un singur tip de strategie, ori până când am atins un număr maxim de iterații, pe care l-am lăsat la valoarea de 10.

Observație: Când într-un turneu cu eliminare concurează doi indivizi ce au aceeași strategie deterministă, la finalul unei iterații, cei doi indivizi vor avea exact același scor. Nu putem spune același lucru despre doi indivizi care folosesc strategia **Random**.

Pentru a vedea clar modul în care evoluează strategiile în contextul acestui tip de turneu, am implementat o metodă grafică de vizualizare a datelor. Am ales să folosesc **line chart**-uri. Axa absciselor are drept legendă numărul de indivizi din fiecare strategie. Axa ordonatelor reprezintă indexul rundei turneului.

În fiecare turneu cu eliminare pot participa 25 de jucători, câte 5 din fiecare strategie. La fiecare turneu cu eliminare sunt câte 5 jucători care urmează exact aceeași strategie obținută în urma rulării algoritmului genetic.

3 Concluzii trase in urma finalizarii turneelor

3.1 Numărul de runde al meciurilor din turneul cu eliminare

Când numărul de runde al merciurilor din turneul cu eliminare este mic, am observat că strategiile propuse de algoritmul genetic, cu mici excepții, nu reușesc să câștige. Numărul de cromozomi, în cele mai optimiste cazuri, înregistrează o creștere și apoi o descreștere până la 0 la final de turneu.

În exemplele de mai jos, configurația algoritmului genetic este următoarea:

- numărul de runde jucate în fiecare meci: 20
- numărul de generații: 500
- dimensiunea populației antrenate: 25
- probabilitatea de încrucișare: 0.1
- probabilitatea de mutație: 0.2
- configurația populației de antrenament:

```
{
        "Always Cooperate": 1,
        "Always Defect": 1,
        "Grudger": 1,
        "Pavlov": 1,
        "Tit-For-Tat": 1,
        "Suspicious Tit-For-Tat": 1,
        "Tit-For-Two-Tats": 1
}
```

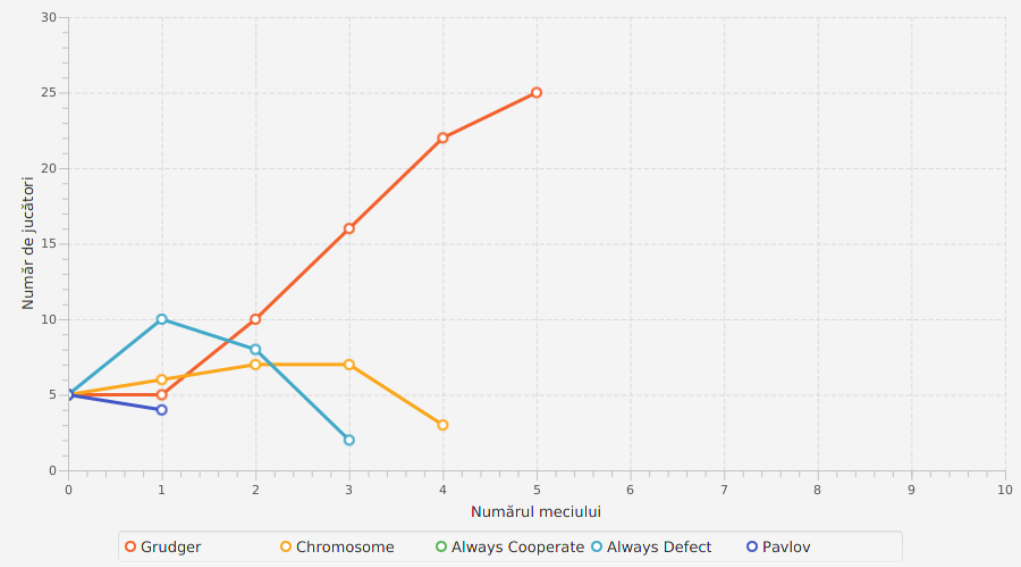

Figura 7: *Evoluția turneului cu eliminare când numărul de runde este 5.*

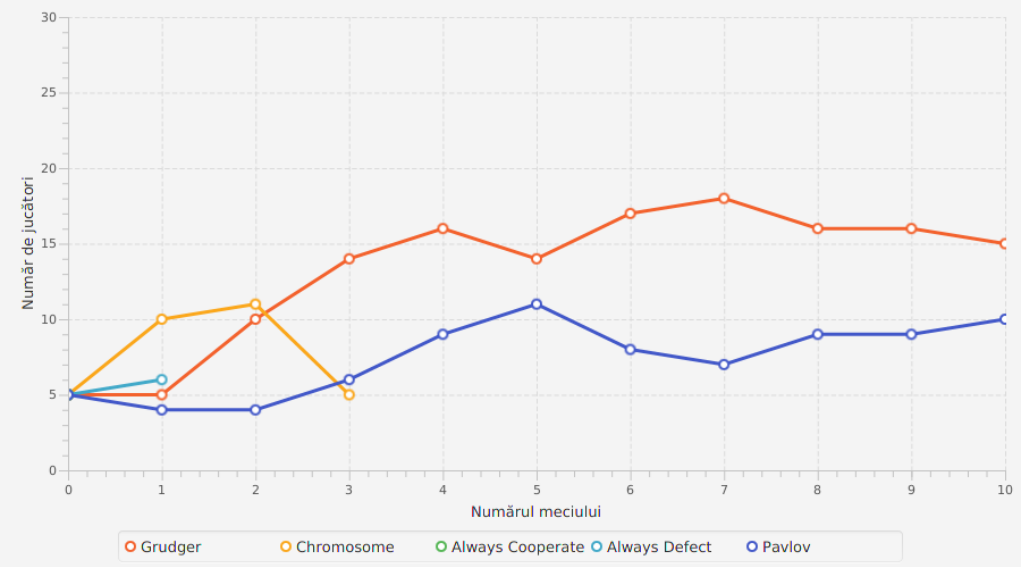

Figura 8: *Evoluția turneului cu eliminare când numărul de runde este 15.*

Mărind cu 2 numărul de runde din fiecare meci, observăm că strategia propusă de algoritmul genetic câștigă.

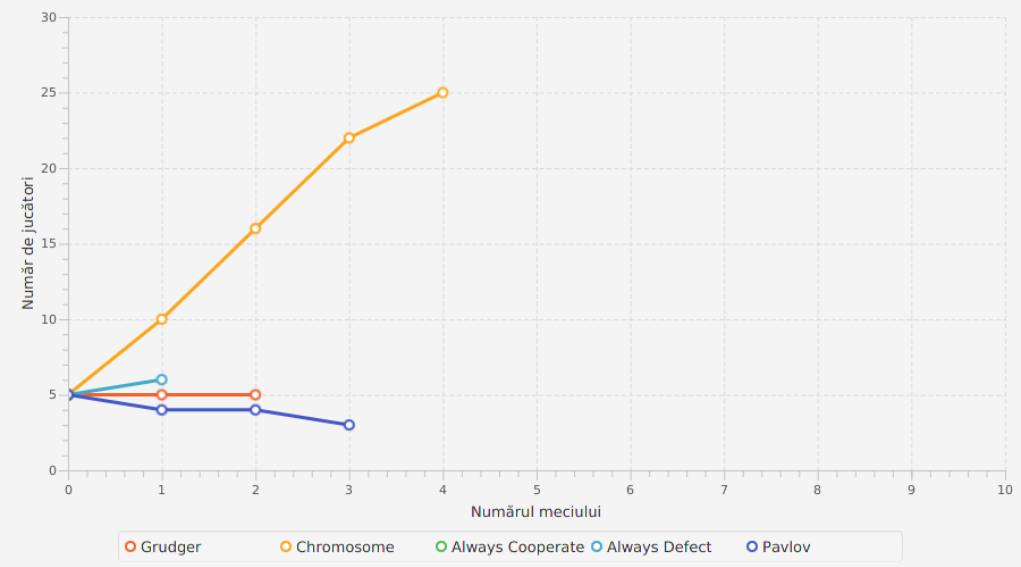

Figura 9: *Evoluția turneului cu eliminare când numărul de runde este 17.*

Dacă numărul de runde al turneului cu eliminare este mai mare, strategia algoritmului genetic are șanse mai mari de câștig.

Am ales ca dimensiunea populatiei sa fie mica in comparatie cu numarul de generatii.

3.2 Populatia de antrenament

Consider că **o populație diversificată de antrenament** poate simula condițiile din faza de testare. Din acest motiv am optat pentru ca aceasta să cuprindă fiecare strategie deterministă prezentată la secțiunea de strategii standard (am omis strategia **Random**). Prezența strategiei **Random** în faza de antrenament nu ajută la interpretarea calității soluției. Sunt șanse ca acestă strategie nedeterministă să obțină un scor bun în faza de antrenament, însă să nu facă față în faza de testare.

Am pus câte un individ din fiecare strategie.

Am încercat să antrenez strategiile în contextul în care populația de antrenament era alcătuită doar din **Always Cooperate**, sau doar din **Always**

Defect, sau doar din Tit-For-Tat, însă aceste strategii nu s-au comportat într-un mod notabil în faza de testare.

Un lucru ce se observă ușor este că strategia **Tit-For-Tat** reușește să câștige aproape de fiecare dată când la initiaizarea turneului cu eliminare toate strategiile au ponderi egale si numarul de runde este mare.

Strategia Tit-For-Tât poate totuși fi totuși bătută. Un astfel de context presupune ca strategiile cu care concurează să aleagă ca la fiecare pas să repete aceeași mișcare, asemeni strategiilor Always Defect și Always Cooperate. Motivul eliminării ar fi că la scoruri egale, când se dorește eliminarea unui singur jucător din 2 candidați, alegerea se face la întâmplare. Acest scenariu presupune schimbarea structurii populației de test. Observația este prezentată în exemplul de mai jos:

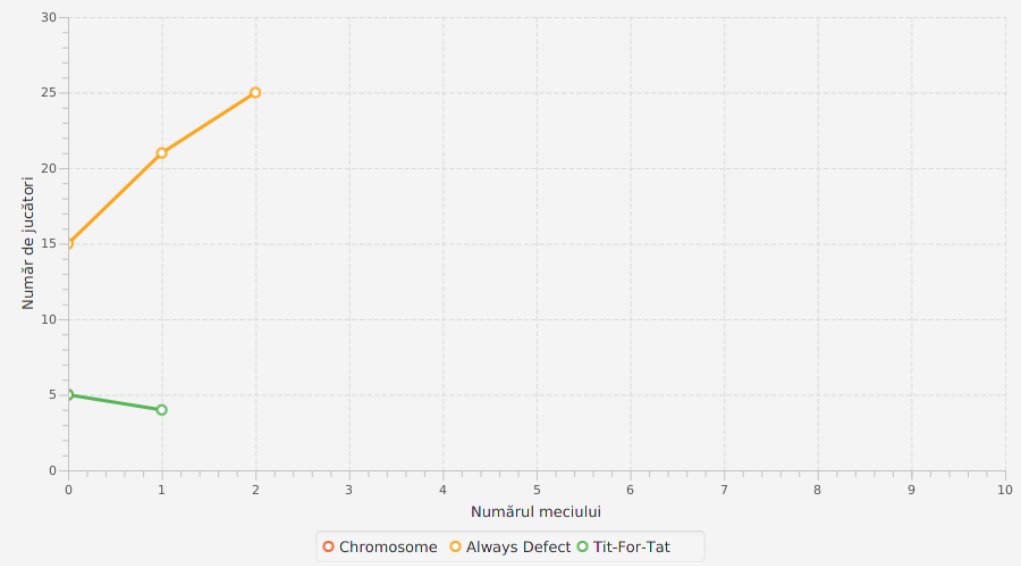

Figura 10: *Strategia* ***Tit-For-Tat*** *pierde.*

Dacă vrem ca în exemplu să includem un cromozom și facem astfel încât strategia Tit-For-Tat să câștige, trebuie să defavorizam cromozomul lăsând numărul de runde la o valoare mică, cum ar fi 5. Dacă ar fi o valoare mai mare, în prima instanță s-ar elimina strategia care repetă aceeași mișcare, ca mai apoi să fie eliminate copiile cromozomilor.

Bibliografie

[1] Merrill M. Flood, Melvin Dresher, Albert W. Tucker, Framing Device, Experimental Economics *Prisoner's Dilemma: Game Theory*. 2010

[2] `https://www.rand.org/`

[3] `https://plato.stanford.edu/entries/prisoner-dilemma/`

[4] `https://dictionary.cambridge.org/dictionary/english/round-robin`

[5] Melanie Mitchell. *An introduction to Genetic Algorithms*

[6] Melanie Mitchell. *Genetic Algorithms: An Overview*. 1995

[7] Robert Axelrod. *The Evolution of Cooperation*. 1984

[8] John Holland. *Adaptation in Natural and Artificial Systems*. 1975/1992

[9] Prof. dr. ing. Gabriel Oltean. *Cursul de Tehnici de inteligenţă computaţională în electronică* . Universitatea Tehnica din Cluj-Napoca
`http://www.bel.utcluj.ro/rom/dce/goltean/tice/10_AlgoritmiGenetici.pdf`
`http://www.bel.utcluj.ro/dce/didactic/tice/tice.htm`

[10] Biblioteca JavaFX:
`http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html`

[11] Biblioteca SLF4J:
`https://www.slf4j.org/`

[12] Biblioteca json-simple:
`https://cliftonlabs.github.io/json-simple/`

[13] Schema algoritm genetic:
goo.gl/YLUE4i
Din lucrarea: Bo Xu, Hongfei Lin, Kavishwar B Wagholikar, Zhihao Yang, Hongfang Liu. *Identifying protein complexes with fuzzy machine learning model*
https://goo.gl/SrAHpD

[14] Prof. dr. Daniela Zaharie:
Cursul de Calcul neuronal și calcul evolutiv . Facultatea de Matematică si Informatică, Universitatea de Vest din Timişoara
goo.gl/hQxKeh

[15] Shashi Mittal, Kalyanmoy Deb. *Optimal Strategies of the Iterated Prisoner's Dilemma Problem for Multiple Conflicting Objectives*

[16] Diagramele au fost realizate cu ajutorul:
https://docs.google.com/presentation
https://docs.google.com/spreadsheets/

[17] Tabelul a fost realizate cu ajutorul:
https://www.tablesgenerator.com/

[18] https://www.latex-project.org/about/
https://tex.stackexchange.com/questions