UNIVERSITATEA „ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Problema iterată a prizonierului. Calitatea soluțiilor oferite de un algoritm genetic în contextul turneelor eliminatorii

propusă de

Denise-Mihaela Goldan

Sesiunea: Iulie, 2018

Coordonator științific

Conf. dr. Adrian Iftene

UNIVERSITATEA „ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Problema iterată a prizonierului. Calitatea soluțiilor oferite de un algoritm genetic în contextul turneelor eliminatorii

Denise-Mihaela Goldan

Sesiunea: Iulie, 2018

Coordonator științific

Conf. dr. Adrian Iftene

Declarație privind originalitatea conținutului lucrării de licență

Avizat,
Îndrumător Lucrare de Licență
Conf. dr. Adrian Iftene
Data ..... Semnătura .....

Subsemnata Goldan Denise-Mihaela, domiciliul în Strada Petru Șchiopul nr.1, bloc L1A, scara A, etaj 1, apartament 3, născută la data de 27.09.1996, identificată prin CNP 2960927226702, absolventă a Universității „Alexandru Ioan Cuza" din Iași, Facultatea de Informatică, promoția 2015-2018, declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că Lucrarea de licență cu titlul „*Problema iterată a prizonierului. Calitatea soluțiilor oferite de un algoritm genetic în contextul turneelor eliminatorii*" elaborată sub îndrumarea dl. Conf. dr. Adrian Iftene, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și, în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ... Semnătură student .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Problema iterată a prizonierului. Calitatea soluțiilor oferite de un algoritm genetic în contextul turneelor eliminatorii*", 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" din 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, 29 Iunie 2018 Absolvent *Denise-Mihaela Goldan*

Introducere

Studiul teoriei jocurilor se resimte în multe domenii. Spre exemplu, în cloud computing, teoria jocurilor poate fi aplicată pentru a modela interacțiunile dintre diverși provideri de servicii cloud, unde scopul clientului este minimizarea costurilor și a pierderilor și maximizarea utilizării resurselor. În domeniul afacerilor, în mod constant, se iau decizii legate de menținerea sau reducerea costurilor unui anumit produs, de crearea unor noi produse, de retragerea lor de pe piață. Aplicând elemente de teoria jocurilor, se pot dezvolta strategii, se pot prezice tendințe și se pot face presupuneri despre modul în care anumite decizii pot influența succesul unei campanii.

Dilema prizonierului este o problemă analizată în teoria jocurilor în care participanții trebuie să aleagă, în mod secret, între a coopera cu celalalt protagonist și a-l trăda și, cu toate că ar avea parte de cel mai mare câștig dacă ar alege cooperarea, neîncrederea în oponent ar putea duce la trădare. Dilema prizonierului este exemplificată în rândurile următoare.

Două firme competitive pun la vânzare același produs. Presupunem că, la un anumit moment, cele două firme iau în calcul, în mod independent, schimbarea prețului produsului. Una din firme dorește să obțină un profit mai mare, așa că reduce prețul produsului, lucru ce poate fi interpetat drept trădarea competitorului. Celălalt întreprinzător alege între a *coopera*, prin menținerea prețului și a *trăda*, prin scăderea prețului, egalând prețul propus de competitor sau oferind un preț și mai mic. O altă variantă ar fi ca cele două firme să lase produsul la același preț. Acest comportament al agenților economici ar însemna împărțirea pieței între aceștia.

Varianta iterată a dilemei prizonierului produce strategii care rezumă tendințele legate de încrederea jucătorilor.

Contribuții

În prezenta lucrare propun studierea stategiilor obținute cu ajutorul algoritmilor genetici pentru varianta iterată a dilemei prizonierului, introducându-le într-un mediu în care populația de jucători este într-o continuă schimbare. În acest scop, am desfășurat o serie de experimente. Cadrul de testare al soluțiilor algoritmului reprezintă un turneu în care, după ce fiecare jucător joacă cu toți ceilalți participanți câte un meci, o parte dintre aceștia, cei mai slabi, este eliminată și o parte, cei mai buni, este duplicată. Este o analogie a faptului că un comportament care duce la succes va fi imitat și unul care duce la eșec își va pierde din popularitate. Odată aleasă o anumită strategie, jucătorul nu o poate modifica în mijlocul turneului. Turneul se încheie atunci când întreaga populație folosește aceeași strategie.

Acest mediu reprezintă un simplu joc, insă, prin exemplul pe care îl oferă, deducem că putem modela noi scenarii, care să respecte criteriile impuse de problema cu care ne confruntăm. În acele scenarii putem introduce interacțiuni între indivizi care iau decizii ținând cont de tendința celorlalți participanți de a coopera sau trăda.

Pentru a trage cele mai bune concluzii, trebuie să facem un studiu prin care să identificăm strategiile adoptate de actorii din problema noastră, să modelăm un mediu de antrenament cu ajutorul unui algoritm genetic și un mediu de testare. În cadrul unor experimente, diverse soluții ale algoritmului genetic vor fi supuse la test, rezultatele putând fi, ulterior, interpretate și propuse pentru rezolvarea problemei.

Cuprins

Problema iterată a prizonierului **9**

1 Enunţul clasic al dilemei prizonierului 9

1.1 Strategii pentru dilema prizonierului 11

2 Varianta iterată a dilemei prizonierului 12

2.1 Strategii pentru problema iterată a prizonierului 13

Algoritm genetic **15**

1 Apariția noțiunii de algoritm genetic 15

2 Schema generală a unui algoritm genetic 16

3 Pseudocod 17

4 Terminologie 17

Dezvoltarea unei strategii pentru problema iterată a prizonierului folosind un algoritm genetic **19**

1 Tehnologii folosite 19

2 Reprezentarea soluției 19

3 Dimensiunea spațiului de căutare 22

4 Funcția de optimizat 22

4.1 Valoarea recompenselor 22

5 Parametrii algoritmului genetic 22

6 Configurația unui algoritm genetic 23
7 Populația de antrenament și de test 23

Turnee eliminatorii între strategii **27**
1 Termeni întâlniţi 27
2 Cum este modelat un turneu cu eliminare 27
3 Configurația unui turneu cu eliminare 28
4 Concluzii trase în urma finalizării turneelor 29
4.1 Explorarea redusă a spațiului de căutare 29
4.2 Numărul de runde al meciurilor din turneul cu eliminare . . 32
4.3 Strategia **Random** 34
4.4 Strategia **Tit-For-Tat** 37

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 închisoare 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.*

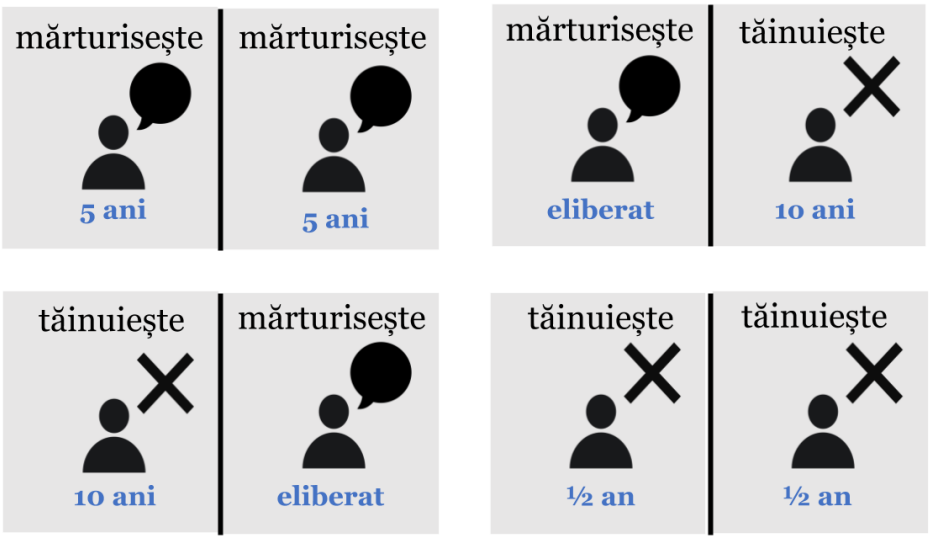

Figura 1: *Prezentare a celor patru posibilități descrise în scenariul anterior.*

Observaţie: *Considerăm că atunci când un suspect mărturisește, mărturia sa îl incriminează 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 mărturisire:

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**

Tabela 1: *Adaptare după matricea recompenselor pentru dilema prizonierului.*

Termenii care apar în tabel sunt următorii:

- **Temptation**: recompensa obținută de jucătorul 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 patru termeni reprezintă scoruri de valoare pozitivă și că scopul suspecților este să obțină un număr cât mai mare de puncte.

Între valorile acestor 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 tăinuim (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 tăinuim (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 tăinuiască 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 jucători 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.

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**.

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

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

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 tournament). 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 [5]. Axelrod a solicitat participanților strategii sub forma 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 altă 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 celălalte două cazuri, alege mișcarea opusă.
- **Random**: Se alege la întâmplare următoarea acțiune.
- **Tit-For-Tat**: Se alege cooperarea la prima rundă. De la runda a doua, jucătorul 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 [5] 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 in 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 (șiruri de caractere sau numere 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 [6], 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 generală a unui algoritm genetic

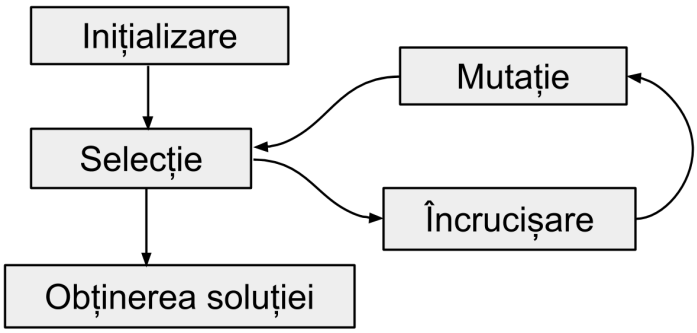

Figura 2: *Schema prezintă componentele de bază ale unui algoritm genetic și legăturile dintre acestea* [7].

3 Pseudocod

```
inițializează cu valori aleatorii populația
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 maxim 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 doi *cromozomi părinți*).

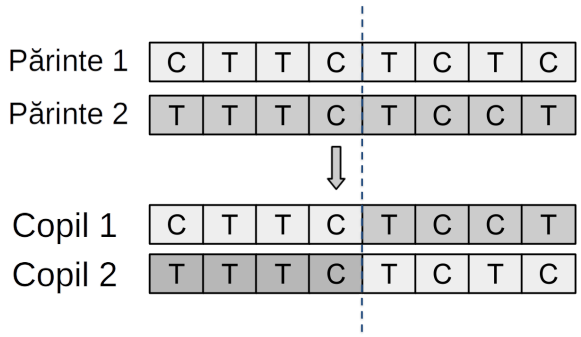

Figura 3: *Exemplu de încrucișare cu un singur punct de tăiere. Punctul de tăiere este marcat prin linia verticală punctată, între elementele de la indexul trei și indexul patru din crmozomi.*

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

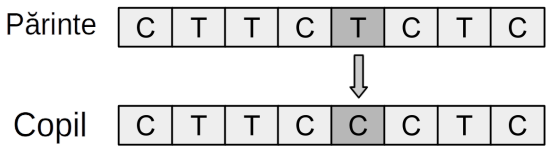

Figura 4: *Mutaţia are loc la poziţia cu indexul patru din cromozomul părinte.*

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 de jurnalizare **SLF4J** [8] [4].

Pentru a vedea cum evoluează strategiile pe parcursul unui turneu cu eliminare, am apelat la biblioteca integrată **JavaFX** [9].

Pentru manipularea fișierelor de tip json, am folosit biblioteca **json-simple** [10].

2 Reprezentarea soluției

În urma organizării celor două turnee, Axelrod [11] a decis să cerceteze dezvoltarea unei strategii pentru problema iterată 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. Concluziile lui Axelrod sunt prezentate 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 la următoarea rundă, cunoscând mișcarea oponentului din runda anterioară.

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:

1. Observăm ce a ales oponentul în runda anterioară.
2. Formăm perechea compusă din mișcarea noastră, împreună cu cea a oponentului.
3. Vedem indexul care corespunde perechii obținute la pasul anterior.
4. 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 șapte poziții de la începutul șirului de caractere au următoarele semnificații:

- La **poziția numărul 1** se găsește mișcarea aleasă pentru prima rundă a jocului;
- **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**);
- **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**);
- **Poziția numărul 4**: mișcarea pe care o vom face la cea de a treia rundă, dacă, pentru primele două runde, istoricul oponentului este **CC**;
- **Poziția numărul 5**: mișcarea pentru cea de a treia rundă, dacă, pentru primele două runde, istoricul oponentului este **CT**;
- **Poziția numărul 6**: mișcarea pentru cea de a treia rundă, dacă, pentru primele două runde, istoricul oponentului este **TC**;
- **Poziția numărul 7**: mișcarea pentru cea de a treia rundă, dacă, la primele două runde, oponentul a avut mișcările **TT**.

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

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 căuta 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.

4.1 Valoarea recompenselor

În implementare, am considerat că **Temptation** are valoarea 5, **Reward** are valoarea 3, **Punishment** are valoarea 1 iar **Sucker's payoff** are valoarea 0 [12].

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 încrucișării, 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

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

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 **training.config.json** și **testing.config.json**. Fișierele pot fi ușor manipulate prin clase specializate în citirea și scrierea lor. În fiecare dintre 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**.

[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 scorurilor obținute de cromozom la fiecare meci.

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 dosar 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 aceeași configurație 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. Numărul de restartări este fixat la valoarea 1000.

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

1. Se aplică un proces de selecție.
2. Se aplică operatorul încrucișării.
3. 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 proporțională 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 proporțională cu gradul de adecvare [13].

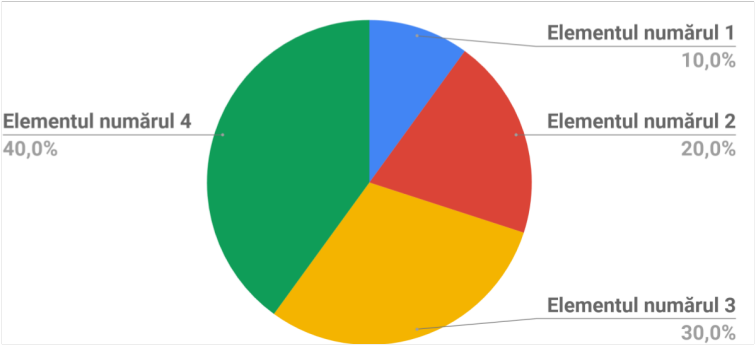

Figura 5: *Probabilități de selecție ale elementelor.*

Figura 5 reprezintă 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%.

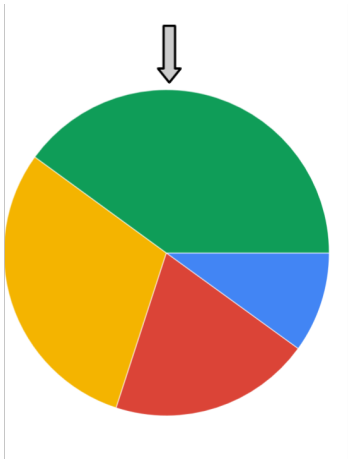

Figura 6: *Rezultatul rotirii ruletei.*

Rezultatul din Figura 6 poate fi interpretat drept selectarea individului care corespunde categoriei etichetate cu verde (se alege elementul cu numărul 4, care a avut o probabilitate de selecție 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 me-

ciurilor, 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 ca 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.

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 numărul meciului din turneu.

3 Configurația unui turneu cu eliminare

Așa cum am stabilit parametrii mediului de antrenare (parametrii algoritmului genetic), va trebui să stabilim și în ce condiții vom testa cromozomii obținuți. Vom impune condiții legate de procentul de jucători care sunt eliminați, respectiv duplicați la final de iterație. De asemenea, vom stabili configurația populației de testare și câte copii după o anumită strategie a algoritmului genetic vor participa la turneul eliminatoriu. În ultimă instanță, vom alege numărul de runde ce va alcătui un meci dintre doi jucători.

Cum s-a discutat în capitolul anterior, populația de testare va putea fi definită în fișierul **testing.config.json**.

În experimentele descrise, am ales ca participanții la turneu să aibă ponderi egale iar procentul de jucători eliminați să fie de 25%.

4 Concluzii trase în urma finalizării turneelor

Fiecare experiment este dat de stabilirea parametrilor configurației algoritmului genetic, urmată de rularea algoritmului genetic și trecerea soluției prin mai multe medii de testare, prin varierea configurației turneului cu eliminare.

În contextul acestei probleme, nu putem vorbi despre optim local sau global, întrucât un fitness cu o valoare bună obținut pentru un cromozom în faza de antrenare poate să își piardă relevanța în faza de testare, datorită faptului că mediul de testare poate varia.

În rândurile următoare prezint o serie de experimente.

4.1 Explorarea redusă a spațiului de căutare

Experimentul 1

Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în ***Anexa numărul 1****.*

Am făcut un prim experiment în care configurația algoritmului genetic a dus la o explorare redusă a spațiului de căutare. Pentru aceasta, am ales valori mici pentru numărul de generații (100), pentru numărul de runde din turneul clasic (10) și pentru probabilitățile operatorilor genetici (10%).

Diagramele de mai jos atestă calitatea redusă a soluției obținute în urma rulării algoritmului genetic.

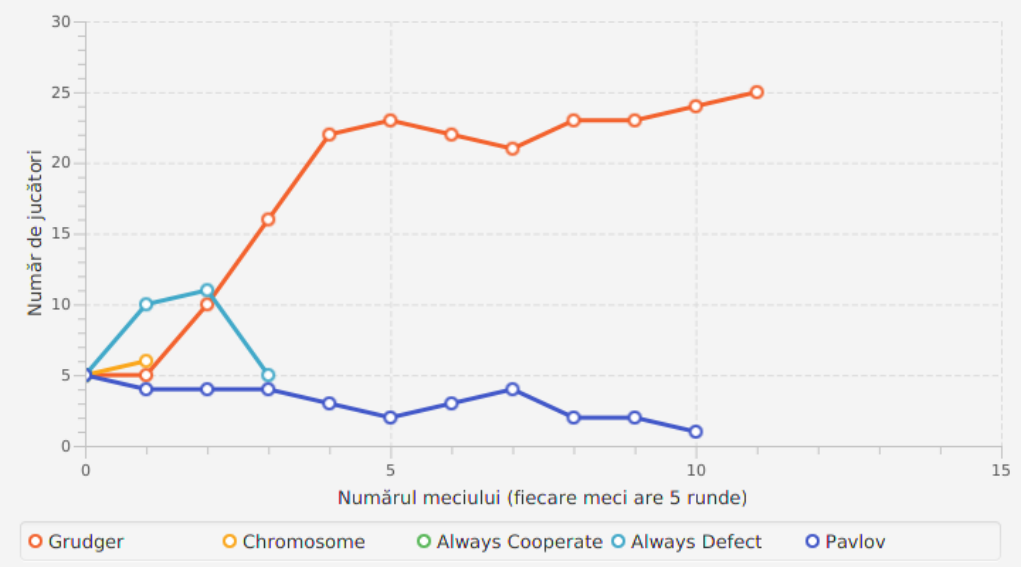

Figura 7: *Cromozomii pierd în turneul cu eliminare cu 5 runde/meci.*

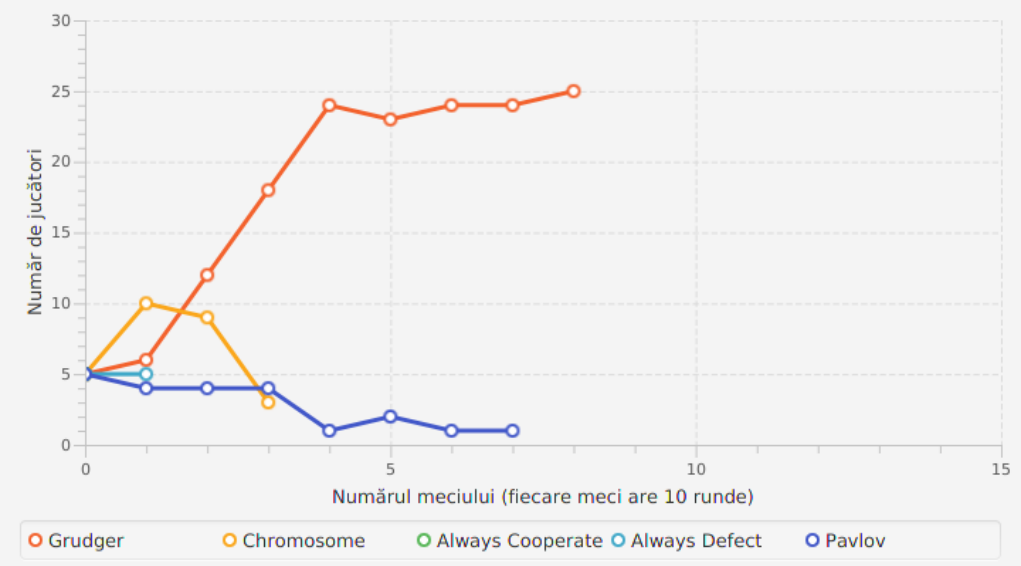

Figura 8: *Cromozomii pierd în turneul cu eliminare cu 10 runde/meci.*

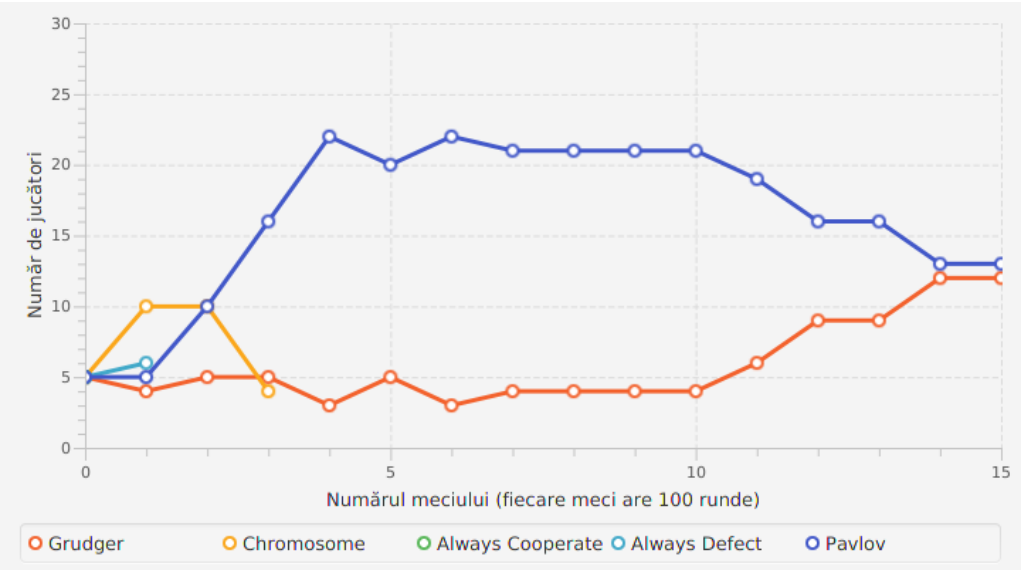

Figura 9: *Cromozomii pierd în turneul cu eliminare cu 100 runde/meci.*

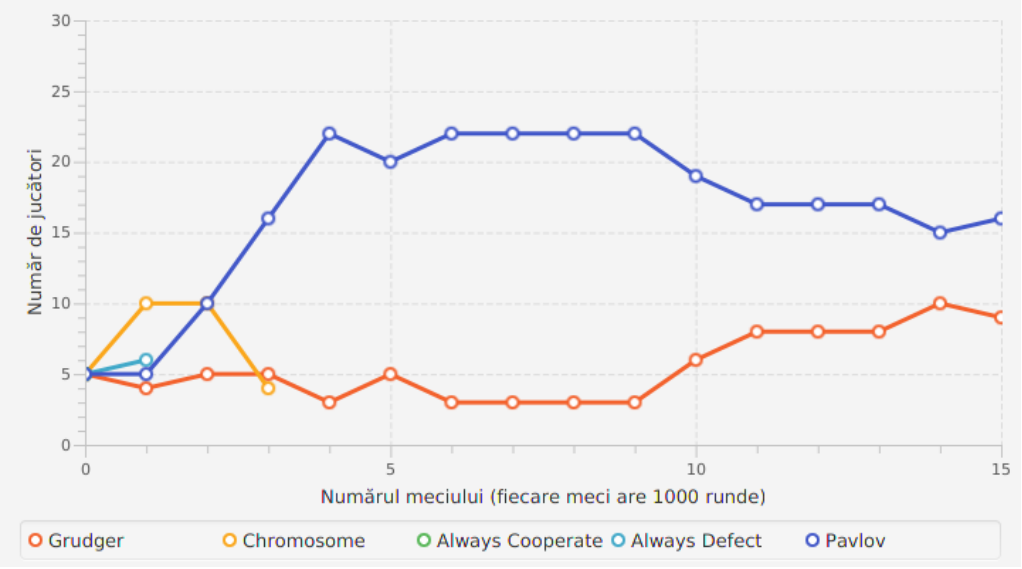

Figura 10: *Cromozomii pierd în turneul cu eliminare cu 1000 runde/meci.*

4.2 Numărul de runde al meciurilor din turneul cu eliminare

Experimentul 2

Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în ***Anexa numărul 2****.*

Într-un alt experiment, am surpins cum numărul de runde al meciurilor din turneul cu eliminare, odată modificat cu o valoare foarte mică, a avantajat una din strategiile algoritmului genetic.

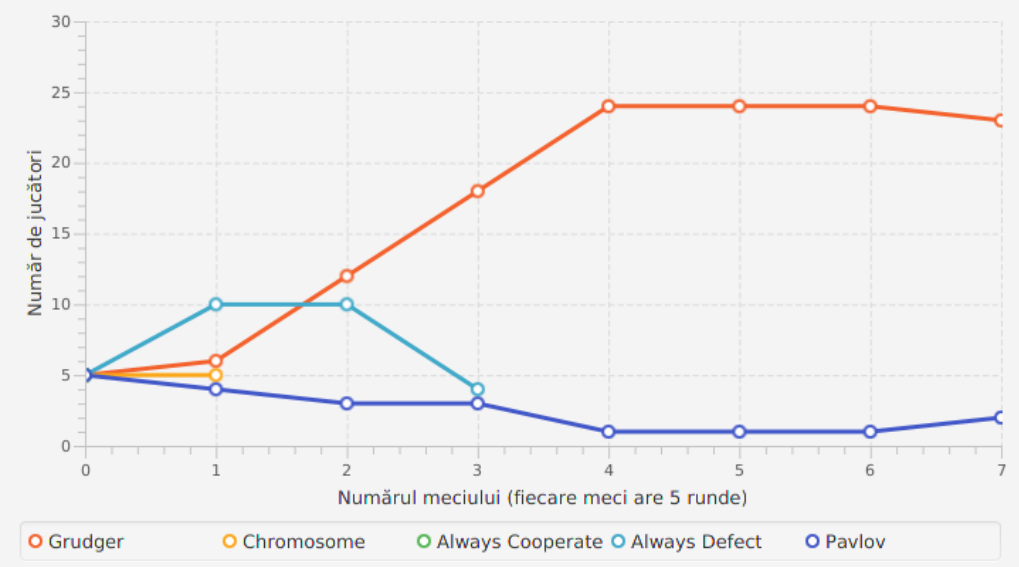

Figura 11: *Cromozomii pierd în turneul cu eliminare cu 5 runde/meci.*

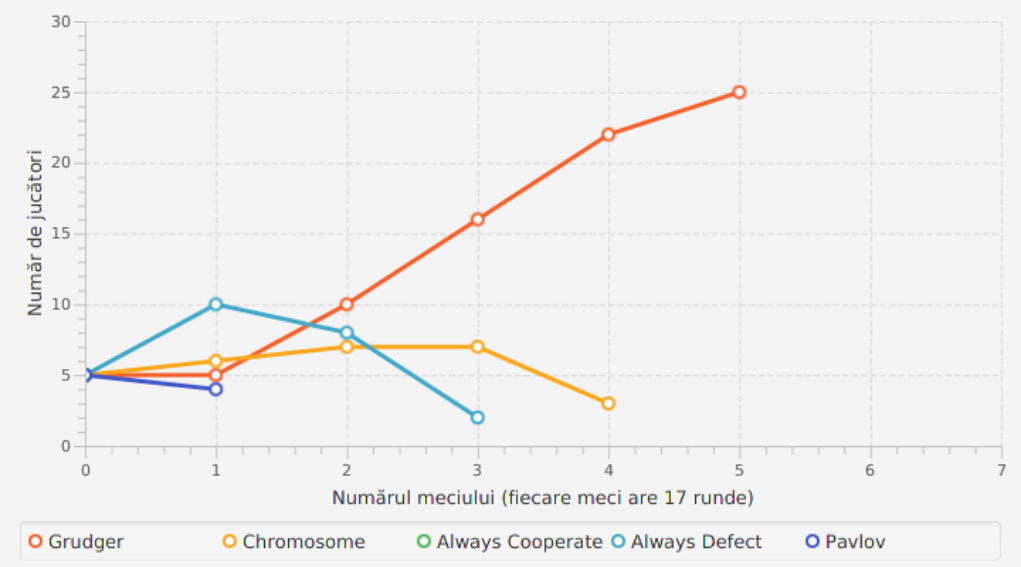

Figura 12: *Cromozomii pierd în turneul cu eliminare cu 17 runde/meci.*

Mărind cu 1 numărul de runde din fiecare meci, observăm că strategia propusă de algoritmul genetic câștigă.

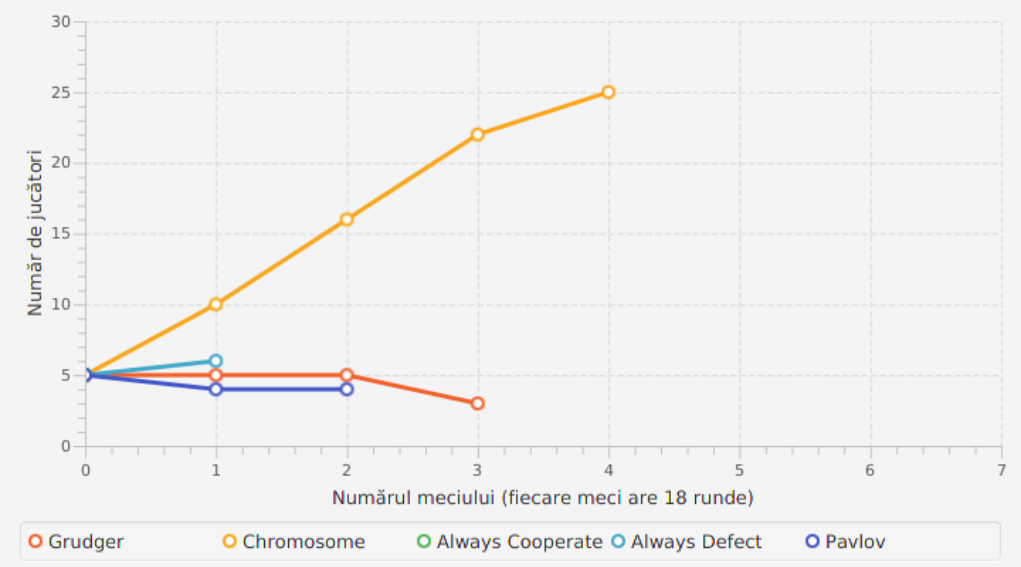

Figura 13: *Cromozomii câștigă în turneul cu eliminare cu 18 runde/meci.*

4.3 Strategia Random

Strategia **Random** nu reprezintă o problemă pentru strategiile oferite de algoritmul genetic.

Am antrenat o populație de dimensiune mică (5 cromozomi) într-un număr mare de generații (1000), lansând valori mari ale mutației și încrucișării (50% pentru ambii operatori). Populația de antrenament este dată de un singur individ de tip **Random**. Am rulat două experimente, în urma cărora am obținut două strategii: pentru prima strategie obținută, cea din experimentul numărul 3, numărul de runde al meciurilor tuneului clasic este mai mic (5 runde); pentru celălalt experiment numărul de runde al meciurilor din turneului clasic este 100.

Experimentul 3

Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în ***Anexa numărul 3****.*

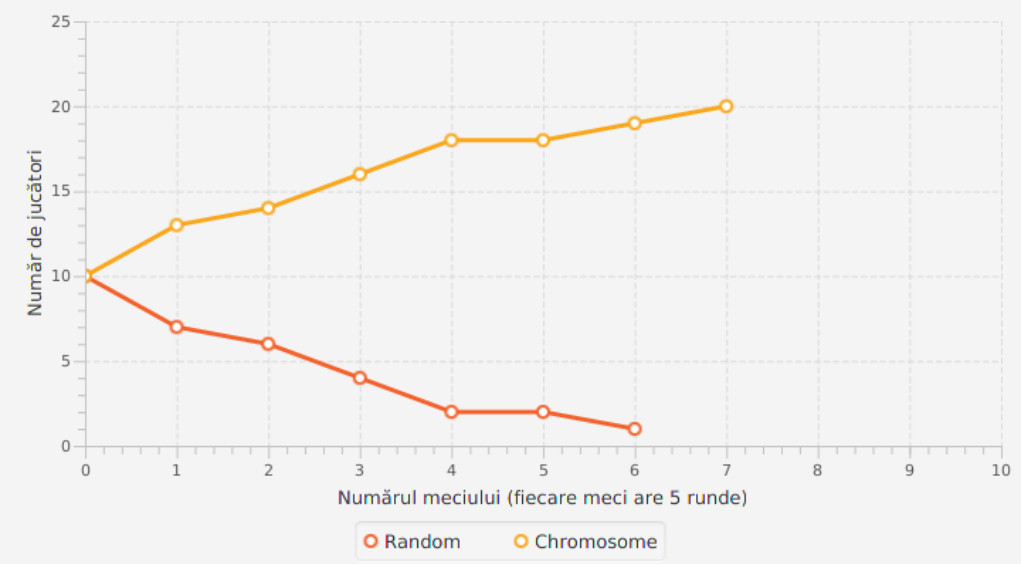

Figura 14: *Cromozomii câștigă în turneul cu eliminare cu 5 runde/meci.*

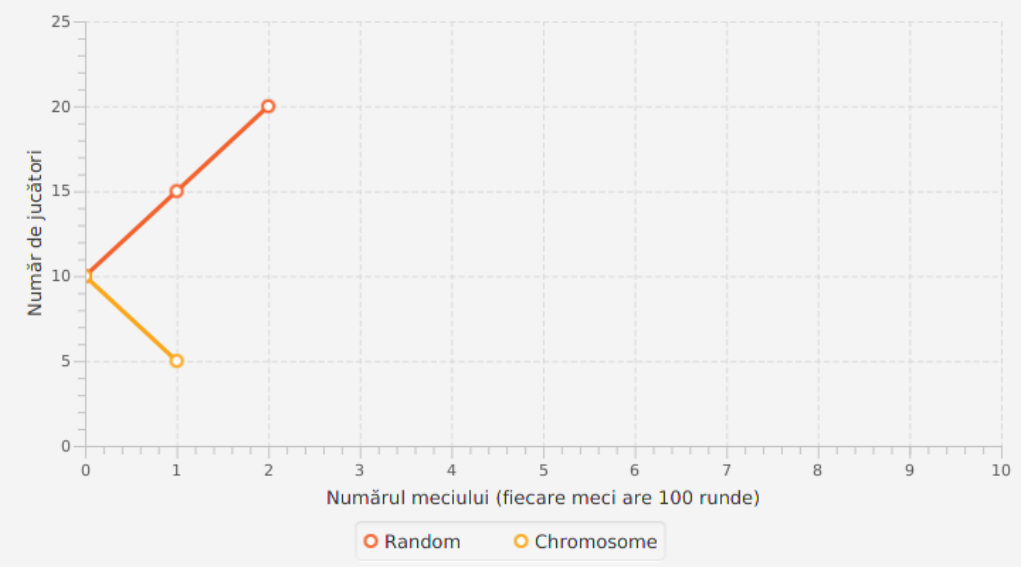

Figura 15: *Strategia propusă de algoritmul genetic pierde, întrucât nu este optimizată pentru a câștiga în contextul unui turneu cu număr mare de runde.*

Experimentul 4

Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în ***Anexa numărul 4****.*

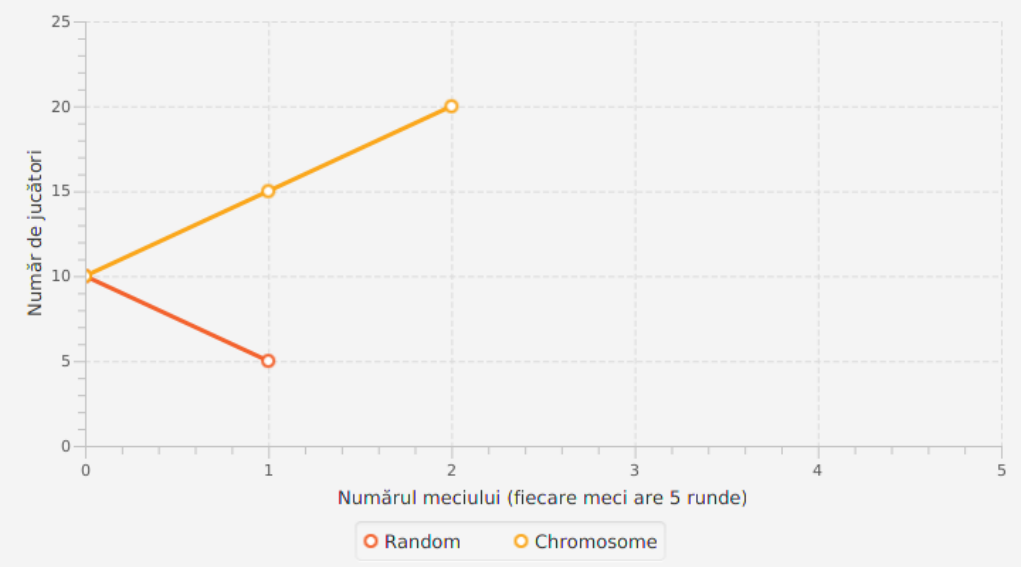

Figura 16: *Cromozomii câștigă în turneul cu eliminare cu 5 runde/meci.*

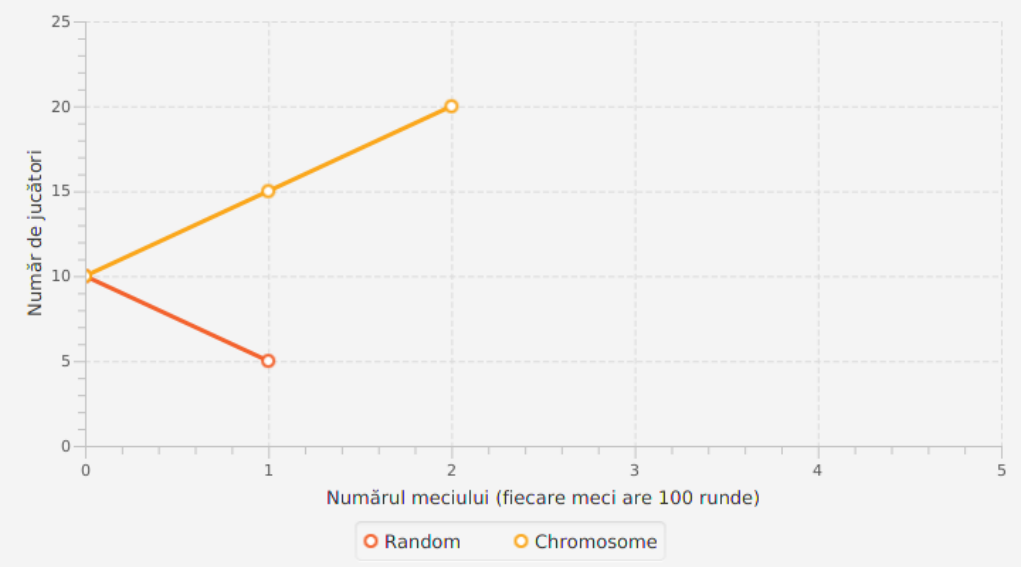

Figura 17: *Cromozomii câștigă în turneul cu eliminare cu 100 runde/meci.*

Strategia din experimentul cu numărul 4 va face față și la un număr de runde mai mic, dar și la un număr de runde mai mare în turneele cu eliminare în fața

strategiei **Random**. Pentru această configurație a turneului eliminatoriu, putem trage concluzia că experimentul numărul 4 oferă o soluție mai bună.

4.4 Strategia Tit-For-Tat

Experimentul numărul 5

Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în ***Anexa numărul 5***.

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

Pentru a bate strategia **Tit-For-Tat** este necesar un număr mic de runde, fiindcă știm că un meci între doi astfel de indivizi le va crește substanțial scorul, întrucât cooperează la fiecare pas. De asemenea, trebuie să ne folosim de faptul că știm că va coopera la prima mișcare. Strategia inamică se va folosi de faptul că atunci când unul cooperează și celălalt trădează, cel care trădează este mai bine răsplătit decât cel ce a cooperat. În alte cuvinte, strategia inamică trebuie să impersoneze strategia **Always Defect**.

Cromozomul, cu eleganța reprezentării sale, poate mima toate strategiile standard descrise în capitolele anterioare. Pentru a imita **Always Defect**, am lăsat ca populația de antrenament să fie dată de un singur individ al acestei strategii, am ales un număr mare de generații (1000), procente mare pentru mutație și încrucișare (50%), o populație de dimensiuni reduse (15).

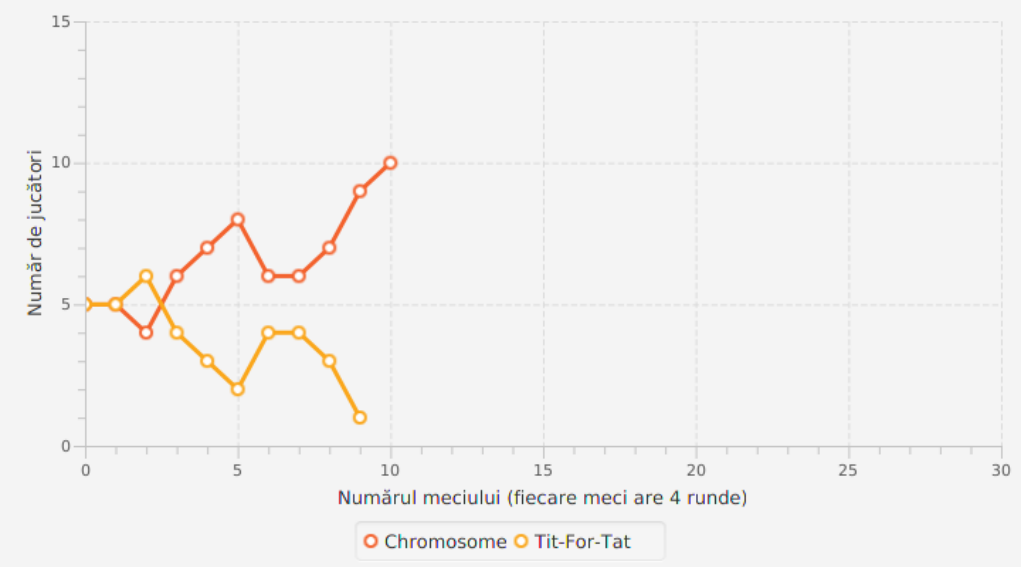

Figura 18: *Cromozomii câștigă turneul cu eliminare în cadrul unui număr mic de runde, câte 4 runde la fiecare meci al turneului cu eliminare.*

Atunci când doi indivizi ce urmează startegia **Tit-For-Tat** concurează într-un meci, vor coopera la fiecare rundă iar cooperarea mutuală este cea mai bine răsplătită, oferind un punctaj pe măsură. Din acest motiv, la un turneu cu eliminare unde numărul de runde este mai mare (mai mare de 5 runde la fiecare meci), va depăși strategiile care trădează în majoritatea rundelor.

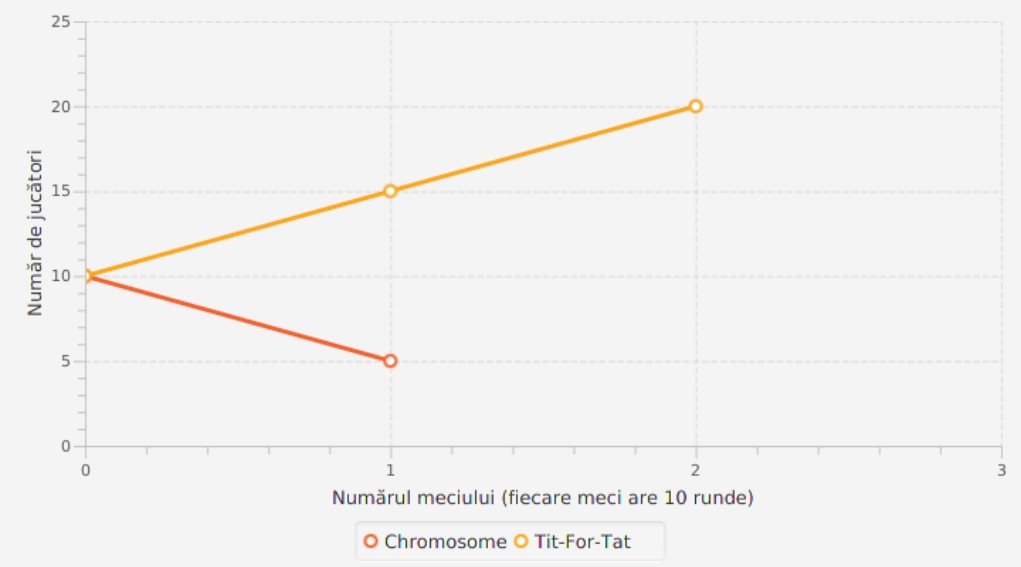

Figura 19: *Strategia **Tit-For-Tat** câștigă când numărul de runde este 10.*

Experimentul numărul 6

*Configurația algoritmului genetic și cea a turneului cu eliminare este trecută în **Anexa numărul 6**.*

Mai există o altă variantă pentru a învinge o populație dată de copii după **Tit-For-Tat**.

Am făcut un experiment în care populația de antrenament este dată de un individ cu strategia **Tit-For-Tat**. Acest lucru a făcut ca strategia oferită de algoritmul genetic să imite strategia **Tit-For-Tat**. În aceste cazuri, cromozomii și copiile după **Tit-For-Tat** vor avea același scor și, la rundele eliminatorii, se vor elimina (și apoi duplica) la întâmplare câteva strategii dintre cele două tipuri. Sunt șanse egale de câştig între cele două tipuri de strategii.

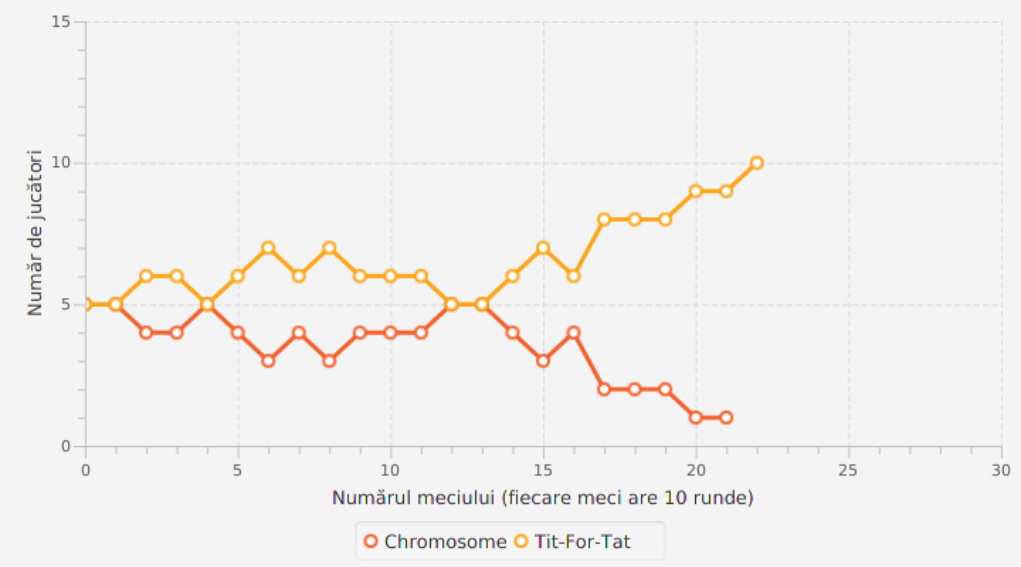

Figura 20: *Populația ce folosește strategia* ***Tit-For-Tat*** *câștigă.*

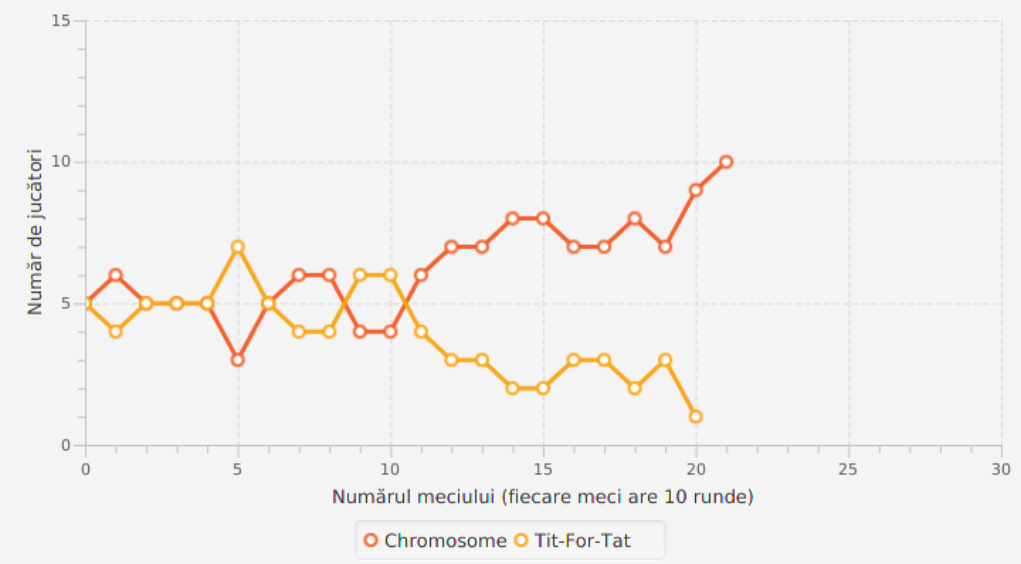

Figura 21: *Cromozomii câștigă.*

Concluziile lucrării

De la începutul proiectului am avut o bănuială cum că nu putem vorbi despre existența unei strategii optime. Această idee este susținută de faptul că utilizatorul poate configura mediul de testare după bunul plac. Schimbarea unui singur parametru poate însemna diferența dintre succes și eșec, așa cum s-a putu observa în experimentul cu numărul 2 (*Numărul de runde al meciurilor din turneul cu eliminare*), în care, incrementând cu 1 numărul de runde/meci, am favorizat populația de cromozomi care, în final, a câștigat turneul cu eliminare. După o serie de experimente, dintre care o parte au fost legate în această lucrare, bănuiala mi-a fost confirmată.

Chiar dacă nu există o strategie care să domine turneul cu eliminare în orice formă a sa, putem folosi algoritmul genetic pentru a modela soluția care să fie optimizată pentru un mediu de testare cunoscut (așa cum s-a putut observa în experimentele desfășurate cu scopul de a învinge strategia **Random** și strategia **Tit-For-Tat**). Argumentul principal al acestei observații este că algoritmii genetici simulează *evoluția* - strategia abordată de speciile biologice pentru a căuta soluții cât mai potrivite, adaptate condițiilor de mediu, într-un număr foarte mare de posibilități.

Cred că această lucrare reprezintă un exemplu ce atestă că putem căuta soluții pentru diverse probleme folosind euristici. Dată fiind o problemă, putem să identificăm actorii, strategiile folosite de aceștia legat de tendințele de cooperare și trădare și putem experimenta cu diverse strategii propuse de o *euristică*. Aceasta favorizează căutarea unei soluții într-un spațiu foarte mare de căutare. Pentru găsirea unei soluții cât mai bune, care să se potrivească mediului de testare, trebuie să derulăm un număr mare de experimente și trebuie să analizăm rezultatele.

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. *Genetic Algorithms: An Overview*. 1995

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

[7] Schemă 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`

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

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

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

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

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

[13] 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

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

[15] https://ncase.me/trust//

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

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

[18] Tabelul a fost realizat cu ajutorul:
https://www.tablesgenerator.com/

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

Anexa numărul 1 - Experimentul numărul 1

Configurația algoritmului genetic pentru primul experiment este următoarea:

- numărul de runde jucate în fiecare meci: 10;
- numărul de generații: 100;
- dimensiunea populației antrenate: 10;
- probabilitatea de încrucișare: 0.10;
- probabilitatea de mutație: 0.10;
- 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
}
```

Configurația turneului cu eliminare:

În turneul cu eliminare au participat câte 25 de indivizi, dintre care 5 sunt copii după cromozomul obținut de algoritmul genetic.

- procentul de jucători care se elimină (și, implicit, procentul de jucători care se duplică) este de 25%;
- configurația populației de testare:

```
{
        "Always Cooperate": 5,
        "Always Defect": 5,
        "Grudger": 5,
        "Pavlov": 5
}
```

Anexa numărul 2 - Experimentul numărul 2

Configurația algoritmului genetic pentru cel de al doilea experiment este următoarea:

- numărul de runde jucate în fiecare meci: 20;
- numărul de generații: 500;
- dimensiunea populației antrenate: 25;
- probabilitatea de încrucișare: 0.10;
- probabilitatea de mutație: 0.20;
- 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
}
```

Configurația turneului cu eliminare:

Este aceeași configurație ca cea prezentată în **Anexa numărul 1**.

Anexa numărul 3 - Experimentul numărul 3

Configurația algoritmului genetic pentru cel de al treilea experiment este următoarea:

- numărul de runde jucate în fiecare meci: 5;
- numărul de generații: 1000;
- dimensiunea populației antrenate: 15;
- probabilitatea de încrucișare: 0.50;
- probabilitatea de mutație: 0.50;
- configurația populației de antrenament:

```
{
        "Random" : 1
}
```

Configurația turneului cu eliminare:

În turneul cu eliminare au participat 20 de indivizi, dintre care 10 sunt copii după cromozomul obținut de algoritmul genetic.

- procentul de jucători care se elimină (și, implicit, procentul de jucători care se duplică) este de 25%;
- configurația populației de testare:

```
{
        "Random" : 10
}
```

Anexa numărul 4 - Experimentul numărul 4

Configurația algoritmului genetic pentru cel de al patrulea experiment este următoarea:

- numărul de runde jucate în fiecare meci: 100;
- numărul de generații: 1000;
- dimensiunea populației antrenate: 15;
- probabilitatea de încrucișare: 0.50;
- probabilitatea de mutație: 0.50;
- configurația populației de antrenament:

```
{
        "Random" : 1
}
```

Configurația turneului cu eliminare:

În turneul cu eliminare au participat 20 de indivizi, dintre care 10 sunt copii după cromozomul obținut de algoritmul genetic.

- procentul de jucători care se elimină (și, implicit, procentul de jucători care se duplică) este de 25%;
- configurația populației de testare:

```
{
        "Random" : 10
}
```

Anexa numărul 5 - Experimentul numărul 5

Configurația algoritmului genetic pentru cel de al cincilea experiment este următoarea:

- numărul de runde jucate în fiecare meci: 100;
- numărul de generații: 1000;
- dimensiunea populației antrenate: 15;
- probabilitatea de încrucișare: 0.50;
- probabilitatea de mutație: 0.50;
- configurația populației de antrenament:

```
{
        "Always Defect": 1
}
```

Configurația turneului cu eliminare pentru experimentul numărul 5:

În turneul cu eliminare au participat 20 de indivizi, dintre care 10 sunt copii după cromozomul obținut de algoritmul genetic.

- procentul de jucători care se elimină (și, implicit, procentul de jucători care se duplică) este de 25%;
- configurația populației de testare:

```
{
        "Tit-For-Tat": 10
}
```

Anexa numărul 6 - Experimentul numărul 6

Configurația algoritmului genetic pentru cel de al șaselea experiment este următoarea:

- numărul de runde jucate în fiecare meci: 100;
- numărul de generații: 1000;
- dimensiunea populației antrenate: 15;
- probabilitatea de încrucișare: 0.50;
- probabilitatea de mutație: 0.50;
- configurația populației de antrenament:

```
{
        "Tit-For-Tat": 1
}
```

Configurația turneului cu eliminare pentru experimentul numărul 6:

În turneul cu eliminare au participat 10 indivizi, dintre care 5 sunt copii după cromozomul obținut de algoritmul genetic.

- procentul de jucători care se elimină (și, implicit, procentul de jucători care se duplică) este de 25%;
- configurația populației de testare:

```
{
        "Tit-For-Tat": 5
}
```