Automatizarea deciziilor agenților dintr-un mediu virtual

Uicoabă Alexandru   
*Facultatea de Electronică, Telecomunicații și Tehnologii Informaționale  
Universitatea Politehnica Timișora*Timișoara, România   
alexandru.uicoaba@student.upt.ro

*Rezumat*—În această lucrare, se propune o analiză a modului în care agenții inteligenți sunt implementați în medii virtuale. Jocurile video au rolul de a introduce utilizatorul într-o lume virtuală. Diversitatea acestor lumi virtuale este una uriașă, fie că ne referim la jocuri de societate, de acțiune, de curse sau de strategie, având posibilitatea de a juca în modul *single player* sau *multi player*. În urma rezultatelor se observă faptul că balanța inclină spre folosirea de algoritmi ce se bazează pe rețele neuronale. De asemenea, majoritatea agenților inteligenți performează foarte bine doar într-un singur mediu virtual.

Cuvinte Cheie—inteligență artificială, agenți inteligenți, jocuri video, machine learning, rețele neuronale

# Introducere

Jocurile video au o istorie destul de îndelungată, de la primele jocurile dezvoltate pe consola **Atari**, ajungând la cunoscutele **Pac-Man**, **Super Mario** și **Sonic**, și până la cele contemporane, precum **Cyberpunk 2077**.

Agenții inteligenți sunt sisteme informatice care sunt plasați în diverse medii virtuale, în cazul de față, în lumi virtuale, care sunt capabili de acțiuni autonome în vederea îndeplinirii unor obiective bine definite.[1] Aceștia au ca scop rezolvarea problemelor din joc, care ar putea fi reprezentate de înfrângerea unui adversar în luptă sau navigarea printr-un labirint.[2]

Lumile virtuale din interiorul jocurilor video reprezintă un teren de joacă pentru o varietate mare de algoritmi, prin intermediul cărora se încearcă lucruri noi. În funcție de rezultatele obținute, aceste idei pot fi transferate și aplicate în viața reală. Scopul inteligenței artificiale este să creeze agenți inteligenți care să fie capabil să ia cele mai bune decizii, atât în mediile virtuale, cât și în lumea reală.[3]

În vederea antrenării acestor agenți, se pot utiliza jocuri deja existente, sau se pot crea jocuri de la zero, într-o multitudine de limbaje de programare, pentru a crea mediul de care avem nevoie.

În ultimul deceniu, s-au realizat din ce în ce mai multe lucrări în această direcție, datorită avansului tehnologic în care ne aflăm, fiind vorba de componente hardware avansate pentru utilizarea algoritmilor dezvoltați care au nevoie de putere mare de calcul. Astfel, au apărut jocuri în care grafica este din ce în ce mai aproape de realitate.

Așadar, prin intermediul agenților inteligenți, se pune accentul pe obținerea unui comportament mai natural al caracterelor din jocuri, prin folosirea de **fuzzy logic** și rețele neuronale. Astfel, se obțin caractere care se comportă natural și care se încadrează în decorul jocului într-un mod mai plăcut.[4]

Astfel, se ajung la următoarele întrebări de cercetare:

Întrebarea 1 – Care sunt tehnicile de învățare automată folosite pentru agenți?

Întrebarea 2 – Pentru ce platformă de jocuri (PC, Console, Web, Mobile) este testat scenariul de agenți inteligenți?

Întrebarea 3 – Care este tipul de interacțiune între agenți (Multi Agents, Single Agent)?

Întrebarea 4 – Agenți sunt creați pentru a juca unul sau mai multe jocuri?

# Metodologie

Această lucrare este rezultatul unei căutări sistematice asupra modului în care se realizează automatizarea deciziilor agenților dintr-un mediu virtual, cum ar fi jocurile video. Această lucrare urmărește metodologia introdusă de Petersen pentru realizarea unui studiu bibliografic.[5]

## Preluarea Datelor

Această analiză se bazează pe articole obținute din baza de date electronică a celor de la IEEE.

Termenii căutați au fost: “intelligent agents in games”, “artificial intelligence agents”, “agents in video games”.

Articolele care se iau în considerare sunt cele care au fost publicate în ultimul deceniu, adică în perioada 2010 – 2020. De asemenea, au fost incluse doar cele care conțin tipul inteligentei artificiale utilizate în vederea automatizării deciziilor agenților. Au fost incluse articolele care sunt realizate în limba engleză și care sunt ușor de analizat.

## Clasificarea articolelor

Extragerea Informațiilor: informația extrasă din articolele incluse a fost organizată în următoarele categorii.

#### Tipul de interacțiune dintre agenți – Este de interes dacă în articole s-au folosit single agents sau multi agents.

#### Tipul de algoritm folosit – Se va analiza dacă în articolul folosit s-a folosit rețele neuronale sau nu.

#### Platforma pe care au fost realizate experimentele

# Rezultate

În această secțiune se vor analiza atât tipul de interacțiune dintre agenți, cât și tipurile de algoritmi folosiți în articolele incluse. Așadar, avem două mari categorii: Single Agent și Multi Agents.

Fig. 1. Repartizarea tipurilor de agenți studiați în articole

Fig. 2. Articolele incluse în această revizuire pe ani

## Single Agent

După cum se observă în Fig. 1,conceptul de Single Agent se regăsește în 9 articole, din cele 21 incluse. În aceste articole au fost automatizați agenți pentru a concura împotriva jocului în sine, cum este de exemplu cazul lui **Pac-Man**, în care acesta trebuie să se ferească de fantome și să mănânce cât mai multe puncte.[6]

Din punctul de vedere al implementării algoritmilor prin care single agentul ia deciziile, există două mari categorii: algoritmi care au la bază rețele neuronale și algoritmi care nu se bazează pe rețele neuronale.

### Algortimi care au la bază rețele neuronale

Din articolele incluse care implementează Single Agent, 7 dintre acestea se bazează pe Rețele Neuronale.

**Monte-Carlo Tree Search** (MTCS) a fost implementat pentru a-l învăța pe **Pac-Man** cum să joace și cum să obțină un scor cât mai mare. În articolul realizat de către Tom Pepels, este foarte importantă consistența. Aceasta reiese din faptul că agentul a obținut un scor ridicat, concurând atât contra unor echipe de fantome mai slabe, cât și împotriva unora mai puternice. Fiind un joc pe mai multe nivele, agentul trebuie să țină cont atât de obiectivele pe termen scurt (short-term goals) cât și de cele pe termen lung (long-term goals).[6]

Un concept în care se folosește MTCS pentru antrenarea agenților este reprezentat de framework-ul **General Video Game Artificial Intelligence** (GVGAI) în cadrul căruia agenții trebuie să joace mai multe jocuri pe care nu le-au mai jucat înainte, dar și să creeze nivele și reguli noi. O limitare sesizabilă cu care se confruntă această platformă este obținerea unui procent de victorie mai mare de 25%, un factor fiind numărul mare de jocuri single player, care depășește 100.[7]

Un alt articol în care s-a implementat un agent care să joace mai multe jocuri diferite este cel al lui Chiara F. Sironi, în care utilizând algoritmul **Self-adaptive Monte-Carlo Tree Search** (SA-MCTS) se ajunge la o versiune extinsă prin reglarea parametrilor on-line (online parameter tuning). Totuși, această versiune oferă o performanță ridicată doar în unele jocuri, unde se poate ajunge la un număr mai mare de simulări. [8]

Un alt joc în care s-a implementat un agent prin intermediul aceluiași algoritm MTCS este **AI Factory Spades**. Folosind acest algoritm se dorește controlarea agenților care nu sunt controlați de către utilizator (aliații și inamicii acestuia), care să emuleze comportamentul uman, păstrându-se nivelul de performanță și competiție. Principiile aplicate în acest joc pot fi transferate către alte jocuri care folosesc algoritmul MTCS.[9]

O altă tehnică de machine learning care are la baza rețelele neuronale este **Reinforcement Learning** (RL). Acesta este prezent în două articole incluse, în care agentul este de tip Single Agent.

Un prim articol în care apare un astfel de algoritm, este cel realizat de către Yuji Kanagawa și Tomoyuki Kaneko. Mai exact, se folosește **Proximity Policy Optimization** (PPO), utilizând mai multe temnițe generate în mod aleatoriu. Rezultatele indică faptul că performanțele au fost mai bune în faza de învățare a agentului, însă scorul general obținut în temnițe noi pentru agent au fost mult sub scorul obținut în faza de învățare.[10]

În cel de-al doilea articol, care este realizat de către Ibrahim Fathy, ni se prezintă un agent inteligent bazat atât pe RL cât și pe **Online Case-Based Planing** (OLCBP), acesta fiind evaluat prin intermediul jocului **Wargus**. Prin hibridizarea celor doi algoritmi, alegerea planurilor de către agent se face într-un mod mai eficient și cu o rată ridicată de succes.[11]

Pac-Man nu este singurul joc clasic în care s-a realizat învățarea automată a agentului. Zpepei Wei și echipa sa au obținut un agent autonom capabil să joace bine cunoscutul joc **Snake**. Antrenarea agentului s-a realizat prin folosirea de **Convolutional Neural Network** (CNN) cu o varietate de Q-learning. Această variantă are ca și scop evitarea situațiilor în care recompensele sunt rare și vin cu întârziere. Experimentele au arătat că prin intermediul acestui model, agentul a reușit chiar să depășească nivelul atins de oameni.[12]

### Algoritmi care nu au la bază rețele neuronale

Sunt prezente 2 articole în care agenți au fost implementați cu algoritmi care nu folosesc rețele neuronale. Ambele articole folosesc **algoritmul Minimax**. Primul articol pune în evidență utilizarea acestui algoritm prin învățarea agentului să joace **Ntxuva**, care este un joc de societate jucat mai ales în Mozambique. Prin utilizarea acestui algoritm, agentul ajunge să joace Ntxuva la un nivel asemănător ca al unui jucător profesionist.[13]

În celălalt articol, prin intermediul unui joc de cărți, se prezintă o variantă îmbunătățită a Minimax-ului printr-o combinație cu **Multi Criteri Decision Maker** (MCDM). Prin folosirea doar a algoritmului MiniMax agentul de tip **Non-Player Character** (NPC) contestă adversarul luând în considerare doar cărțile convocate de către acesta. Prin combinația cu MCDM, agentul ia în considerare și alte informații, cum ar fi valoarea vieții a inamicului și a lui, costul convocării unei anumite cărți, etc. Așadar, agentul va putea să optimizeze daunele produse ca acestea să fie cât mai mari. Astfel, se ajunge la o dificultate crescută, deoarece strategiile generate de către agent sunt mult mai variate.[14]

## Multi Agents

Tabelul 2 înfățișează că în 12 din cele 21 de articole incluse s-a abordat conceptul de Multi Agents. Multi Agents înseamnă folosirea a mai multor agenți inteligenți care împart același mediu virtual.

La fel ca în cadrul single agent, agenții au fost antrenați folosind algoritmi care se bazează pe rețele neuronale și care nu au la bază rețele neuronale.

### Algortimi care au la bază rețele neuronale

Pornind de la RL, pentru ca agenții să rezolve task-urile împreună și în vederea obținerii unui caștig maximal în jocuri repetitive, Zhen Zhang propune algoritmul **Probability of Maximal Reward based on the Infinitesimal Gradient Ascent** (PMR-IGA). Un astfel de rezultat se obține indiferent de condițiile inițiale, având un număr finit de jucători și un număr finit de acțiuni. Folosind o astfel de abordare, ne permite abordarea unor probleme mai practice. În articol, algoritmul prezintă rezultate bune și o consistență în scenariile stocastice.[15]

Articolul realizat de către Damon Daylamani-Zad și Marios C. Anfelides se focusează pe folosirea **Deep Reinforcement Learning** (DRL), cu precădere a PPO, cu scopul de a descoperii impactul pe care îl are altruismul și egoismul asupra credibilității agenților. În urma experimentelor, jucătorii găsesc comportamentul altruist mai credibil decât cel egoism.[16]

Tabelul 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Nume articol** | **Tehnica de învățare** | **Joc** | **Platformă** |
| Real-Time Monte Carlo Tree Search in Ms Pac-Man | Monte Carlo Tree Search | single game | - |
| Analysis of Self-Adaptive Monte Carlo Tree Search in General Video Game Playing | Self-adaptive Monte-Carlo Tree Search  N-Tuple Bandit Evolutionary Algorithm | multi games | - |
| General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms | Monte-Carlo Tree Search | multi games | - |
| Autonomous Agents in Snake Game via Deep Reinforcement Learning | Deep Q-learning network | single game | - |
| A MiniMax Agent for Playing Ntxuva Game – The Mozambican Variant of Mancala | Minimax Algorithm | single game | - |
| Improvisation of Minimax Algorithm with Multi Criteria Decision Maker (MCDM) in the Intelligent Agent of Card Battle Game | Minimax Algorithm with Multi Criteria Decision Maker | single game | - |
| Emulating Human Play in a Leading Mobile Card Game | Information Set Monte Carlo Tree Search | single game | - |
| Intelligent online case-based planning agent model for real-time strategy games | Online Case-based Planning Reinforcement Learning | single game | - |
| Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning | Proximal Policy Optimization | single game | - |

Tabelul 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nume articol** | **Tehnica de învățare** | **Joc** | **Platformă** | |
| A Gradient-Based Reinforcement Learning Algorithm for Multiple Cooperative Agents | Reinforcement Learning | multi games | | - |
| Multi-agent system application in accordance with game theory in bi-directional coordination network model | Bi-directional coordination network  Recurrent Neural Networks | - | | - |
| Altruism and Selfishness in Believable Game Agents: Deep Reinforcement Learning in Modified Dictator Games | deep reinforcement learning | single game | | - |
| Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games | Markov Decision Process  Reinforcement Learning | single game | | - |
| Catch me if you can: A pursuit-evasion game with intelligent agents in the Unity 3D game environment | A\* algorithm | single game | | - |
| General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms | Monte-Carlo Tree Search | multi games | | - |
| A Methodology for Creating Generic Game Playing Agents for Board Games | Monte-Carlo Tree Search  Cascade Correlation Neural Network | single game | | - |
| Utilizing Multiple Agents for Decision Making in a Fighting Game | Deep reinforcement learning | single game | | - |
| Evaluating Competition in Training of Deep Reinforcement Learning Agents in First-Person Shooter Games | Deep Reinforcement Learning | single game | | - |
| Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents | Deep Active Imitation  Direct Imitation Learning | single game | | - |
| Reinforcement Learning with an Extended Classifier System in Zero-sum Markov Games | eXtended Classifier System | single game | | - |
| A Multi-agent Design of a Computer Player for Nine Men's Morris Board Game using Deep Reinforcement Learning | Convolutional Neural Network  Monte Carlo Tree Search | single game | | - |
| Learning to Coordinate with Deep Reinforcement Learning in Doubles Pong Game | Deep Q Networks | single game | | - |

În articolul realizat de către Yoshina Takano, s-au implementat doi agenți într-un joc de lupte (figthing game), utilizându-se atât DRL cât și MCST. În cadrul acestui joc, unul din agentul implementat folosind DRL poate fi considerat un agent dublu, deoarece acesta are atât un rol ofensiv, cât și defensiv. S-a realizat o comparație între cei doi, și a rezultat că agentul dublu DRL a avut cu 30% mai multe șanse de câștig ca agentul simplu MCTS.[17]

Tot prin intermediul DRL se pot antrena agenți în jocuri de tipul **First-Person Shooter**. În experimentul realizat de către Paulo B. S. Serafim s-a realizat împărțirea agenților în 3 grupuri de câte 5. Primele două grupuri s-au antrenat concurând una versus celeilalte, iar ultima dintre ele s-a antrenat concurând versus oponenți care se comportă aleatoriu. Din rezultate reiese faptul că agenții care s-au antrenat contra unor agenți autonomi au avut rezultate mult mai bune decât cei care s-au antrenat cu oponenți aleatorii. De asemenea, un lucru interesant este faptul că agenții au dobândit și comportamente foarte complexe, cum ar fi anticiparea mișcărilor adversarului.[18]

Articolul realizat de către Jafar Abukhait folosește o combinație între MTCS și CNN pentru a învăța agentul să joace **Nine Men’s Morris**, acesta fiind un joc de societate. Agenții cunosc doar regulile jocului, fără a avea acces la seturi de date. Agenții prezintă două faze, acestea fiind plasarea și mutarea. Au fost implementați 3 agenți, care au comunicat între ei în toate fazele de desfășurare ale jocului. Doi dintre agenți sunt folosiți pentru fazele *basic* ale jocului, al treilea fiind introdus pentru capturarea pieselor, deoarece acest lucru se face prin strategie, nefăcându-se aleatoriu.[19]

Un altfel de algoritm RL este cel bazat pe **eXtended Classifier System** (XCS). Acesta are rolul de a menține regulile de concurență pentru selecția acțiunilor dar și pentru utilizarea unui algoritm genetic în vederea căutării politicii optimale. Folosind acest algoritm, agentul este capabil să învețe de la 0 doar prin observarea comportamentului agentului oponent. Astfel, procesul de învățare este unul foarte rapid.[20]

În ultimul articol inclus, în care se folosește DRP, ni se prezintă o modalitate de învățare a agenților să joace **Pong**. Astfel se evidențiază felul în care 2 agenți joacă într-un mod cooperativ un joc la dublu de Pong, prin împărțirea sarcinilor. O limitare a acestui experiment este faptul că agenții nu iau nici o decizie atunci când bila se află la granița responsabilităților celor doi agenți.[21]

Pe lângă algoritmul RL, în vederea automatizării deciziilor agențiilor se folosește și **Recurent Neural Networks** (RNNs) împreună cu **Bi-directional coordination network** (BiCNet). În articolul realizat de Zhang Jie se prezintă un concept de loialitate în care se analizează relația dintre câștigurile individuale și cele globale ale agenților. Modelul a fost verificat prin exemple de evitare de obstacole, distribuția focului și acoperire cooperativă. [22]

Un alt concept interesant este evidențiat prin introducerea unei metodologii care poartă numele de UCT-CCNN. Această metodă execută un număr mare de meciuri între agenți MTST care utilizează **Upper Confidence Bounds for Tree** (UCT), pentru generarea unei baze de date care conține stările posibile. Mai apoi se folosește o rețeaua neuronală **Cascade Correlation Neural Network** (CCNN). Acest concept a fost testat prin intermediul a două jocuri, Othello și Nine Men’s Morris. Agenții obținuți au reușit să câștige meciuri împotriva unor agenți creați special pentru aceste jocuri. De asemenea, prin folosirea UCT-CCNN se pot genera agenți inteligenți cu diferite niveluri de dificultate. În plus, în loc de MCST se pot utiliza și alți algoritmi, cum ar fi MiniMax sau **Alpha-Beta Pruning**.[23]

Un ultim algoritm care se bazează pe rețele neuronale este reprezentat de **Deep Learning** (DP). **Strategia Deep Active Imitation** (DAI) a fost utilizată pentru a învăța agenții din mediul virtual **FIFA**. În experiment s-au folosit 2 agenți, unul antrenat folosind DAI iar celălalt folosind **Direct Imitation Learning** (DIL). S-a putut observa că agentul DAI a fost capabil să obțină rezultate mai bune în situații noi, necunoscute anterior. [24]

### Algoritmi care nu au la bază rețele neuronale

Din cele 12 articole în care s-au prezentat concepte despre multi agents, doar în două s-au automatizat agenți care utilizează algoritmi care nu au la bază rețelele neuronale.

Prin intermediul unui joc de urmărire, în care un utilizator care controlează o bilă trebuie să nu fie prins de alte 3 bile, s-a automatizat un agent folosind **algorimul A\***. Pentru una dintre bile s-a dezvoltat un sistem bazat pe **fuzzy logic**, care este capabil să genereze puncte de intercepție pentru a prinde bila controlată de către utilizator. Prin folosirea acestui sistem, s-a crescut performanța în comparație cu sistemul convențional.[25]

Un alt tip de implementare este cea care folosește **Markov Decision Process for Decision Making (MDP)**. Folosind acest algoritm, pornind de la comportamenul albinelor, s-au create grupuri cooperative de agenți inteligenți pentru jocuri de tip **Real-Time Strategy** (RTS). Rezultatele au arătat un nivel ridicat de inteligență colectivă. Un lucru interesant a fost apariția temporară de așa numiți Team Leaders, care au condus cu succes echipa în bătălie.[26]

# Discuții

După cum reiese și din Tabelul 1, cel mai întâlnit algoritm folosit pentru Single Agent este algoritmul MTCS. Acest algoritm a fost folosit și pentru crearea agenților capabili să joace mai multe jocuri, și nu unul singur.

Fig. 3. Tipul de algoritmi utilizați în articole

În ambele cazuri, atât pentru Single Agent, cât și pentru Multi Agents, au fost preferate implementările de rețele neuronale pentru învățarea agenților. Acest lucru reiese din Tabelul 2, și ne sugerează că rețele neuronale sunt un pion important în acest domeniu.

Tabelul 1 și Tabelul 2 indică faptul că mai este nevoie de așteptare pentru a realiza un agent care să obțină rezultate foarte bune în mai multe jocuri. Majoritatea agenților descriși au obținut performanțe foarte bune doar într-un singur joc. În articolele în care s-au prezentat agenți inteligenți care sunt capabili să concureze în mai multe jocuri, numărul acestor medii virtuale a fost unul scăzut, iar conceptul pe care se bazează jocurile au fost asemănător.

### Limitări.

Având în vedere că în acest articol științific au fost introduse doar articole provenite din baza electronică a celor de la IEEE, este posibil ca unele articole să fie omise. Pe viitor se dorește extinderea articolelor analizate, prin introducerea de articole provenite și din alte baze de date electronice, diferite de IEEE.

# Concluzie

În această lucrare, s-a prezentat un studiu referitor la modul în care sunt implementați agenții inteligenți în medii virtuale. În rezultate se poate observa că în majoritatea articolelor incluse, aceștia au fost implementați folosind rețele neuronale. De asemenea, majoritatea agenților au fost utilizați doar într-un singur mediu virtual, cu câteva excepții. Platformele pe care au fost testați aceștia lipsesc, ceea ce poate însemna că, acestea nu au un rol esențial în performanțele agenților, fie că ne referim la jocuri pe calculator, consolă, web sau dispozitive mobile.

Pe viitor se dorește continuarea acestui articol științific, prin implementarea unui agent inteligent într-un joc de curse de mașini, la un nivel ridicat. Jocul respectiv poate să fie unul existent, sau poate să fie creat de la zero. Din punct de vedere al algoritmului, se va utiliza RL, scopul agentului fiind să simuleze o mașină autonomă în diferite situații.

##### Referințe

[1] L. Padgham and M. Winikoff, Developing Intelligent Agent Systems: A Practical Guide, 1st ed. Wiley, 2004.

[2] F. Safadi, R. Fonteneau, and D. Ernst, ‘Artificial intelligence in video games: Towards a unified framework’, Int. J. Comput. Games Technol., vol. 2015, 2015.

[3] D. A. Chentsov and S. A. Belyaev, ‘Monte Carlo Tree Search Modification for Computer Games’, in 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Jan. 2020, pp. 252–255, doi: 10.1109/EIConRus49466.2020.9039281.

[4] F. Dignum, J. Westra, W. A. van Doesburg, and M. Harbers, ‘Games and Agents: Designing Intelligent Gameplay’, Int. J. Comput. Games Technol., vol. 2009, pp. 1–18, 2009, doi: 10.1155/2009/837095.

[5] K. Petersen, S. Vakkalanka, and L. Kuzniarz, ‘Guidelines for conducting systematic mapping studies in software engineering: An update’, Inf. Softw. Technol., vol. 64, pp. 1–18, Aug. 2015, doi: 10.1016/j.infsof.2015.03.007.

[6] T. Pepels, M. H. M. Winands, and M. Lanctot, ‘Real-Time Monte Carlo Tree Search in Ms Pac-Man’, IEEE Trans. Comput. Intell. AI Games, vol. 6, no. 3, pp. 245–257, Sep. 2014, doi: 10.1109/TCIAIG.2013.2291577.

[7] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, ‘General Video Game AI: A Multitrack Framework for Evaluating Agents, Games, and Content Generation Algorithms’, IEEE Trans. Games, vol. 11, no. 3, pp. 195–214, Sep. 2019, doi: 10.1109/TG.2019.2901021.

[8] C. F. Sironi and M. H. M. Winands, ‘Analysis of Self-Adaptive Monte Carlo Tree Search in General Video Game Playing’, in 2018 IEEE Conference on Computational Intelligence and Games (CIG), Aug. 2018, pp. 1–4, doi: 10.1109/CIG.2018.8490402.

[9] H. Baier, A. Sattaur, E. J. Powley, S. Devlin, J. Rollason, and P. I. Cowling, ‘Emulating Human Play in a Leading Mobile Card Game’, IEEE Trans. Games, vol. 11, no. 4, pp. 386–395, Dec. 2019, doi: 10.1109/TG.2018.2835764.

[10] Y. Kanagawa and T. Kaneko, ‘Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning’, in 2019 IEEE Conference on Games (CoG), Aug. 2019, pp. 1–8, doi: 10.1109/CIG.2019.8848075.

[11] I. Fathy, M. Aref, O. Enayet, and A. Al-Ogail, ‘Intelligent online case-based planning agent model for real-time strategy games’, in 2010 10th International Conference on Intelligent Systems Design and Applications, Nov. 2010, pp. 445–450, doi: 10.1109/ISDA.2010.5687225.

[12] Z. Wei, D. Wang, M. Zhang, A. Tan, C. Miao, and Y. Zhou, ‘Autonomous Agents in Snake Game via Deep Reinforcement Learning’, in 2018 IEEE International Conference on Agents (ICA), Jul. 2018, pp. 20–25, doi: 10.1109/AGENTS.2018.8460004.

[13] F. D. M. A. Ali, E. Gimo, and S. M. Saide, ‘A MiniMax Agent for Playing Ntxuva Game – The Mozambican Variant of Mancala’, in 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), Aug. 2020, pp. 1–5, doi: 10.1109/icABCD49160.2020.9183848.

[14] S. D. H. Permana, K. B. Y. Bintoro, B. Arifitama, A. Syahputra, and M. Cendana, ‘Improvisation of Minimax Algorithm with Multi Criteria Decision Maker (MCDM) in the Intelligent Agent of Card Battle Game’, in 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), Oct. 2019, pp. 254–258, doi: 10.1109/ICECOS47637.2019.8984477.

[15] Z. Zhang, D. Wang, D. Zhao, Q. Han, and T. Song, ‘A Gradient-Based Reinforcement Learning Algorithm for Multiple Cooperative Agents’, IEEE Access, vol. 6, pp. 70223–70235, 2018, doi: 10.1109/ACCESS.2018.2878853.

[16] D. Daylamani-Zad and M. C. Angelides, ‘Altruism and Selfishness in Believable Game Agents: Deep Reinforcement Learning in Modified Dictator Games’, IEEE Trans. Games, pp. 1–1, 2020, doi: 10.1109/TG.2020.2989636.

[17] Y. Takano, S. Ito, T. Harada, and R. Thawonmas, ‘Utilizing Multiple Agents for Decision Making in a Fighting Game’, in 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), Oct. 2018, pp. 594–595, doi: 10.1109/GCCE.2018.8574675.

[18] P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. C. Neto, ‘Evaluating Competition in Training of Deep Reinforcement Learning Agents in First-Person Shooter Games’, in 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Oct. 2018, pp. 117–11709, doi: 10.1109/SBGAMES.2018.00023.

[19] J. Abukhait, A. Aljaafreh, and N. Al-Oudat, ‘A Multi-agent Design of a Computer Player for Nine Men’s Morris Board Game using Deep Reinforcement Learning’, in 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Oct. 2019, pp. 489–493, doi: 10.1109/SNAMS.2019.8931879.

[20] C. Wang, H. Chen, C. Yan, and X. Xiang, ‘Reinforcement Learning with an Extended Classifier System in Zero-sum Markov Games’, in 2019 IEEE International Conference on Agents (ICA), Oct. 2019, pp. 44–49, doi: 10.1109/AGENTS.2019.8929148.

[21] E. A. O. Diallo, A. Sugiyama, and T. Sugawara, ‘Learning to Coordinate with Deep Reinforcement Learning in Doubles Pong Game’, in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2017, pp. 14–19, doi: 10.1109/ICMLA.2017.0-184.

[22] J. Zhang, G. Wang, S. Yue, Y. Song, J. Liu, and X. Yao, ‘Multi-agent system application in accordance with game theory in bi-directional coordination network model’, J. Syst. Eng. Electron., vol. 31, no. 2, pp. 279–289, Apr. 2020, doi: 10.23919/JSEE.2020.000006.

[23] M. Rezende and L. Chaimowicz, ‘A Methodology for Creating Generic Game Playing Agents for Board Games’, in 2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Nov. 2017, pp. 19–28, doi: 10.1109/SBGames.2017.00011.

[24] M. P. P. Faria, R. M. S. Julia, and L. B. P. Tomaz, ‘Evaluating the Performance of the Deep Active Imitation Learning Algorithm in the Dynamic Environment of FIFA Player Agents’, in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Dec. 2019, pp. 228–233, doi: 10.1109/ICMLA.2019.00043.

[25] İ. Şahin and T. Kumbasar, ‘Catch me if you can: A pursuit-evasion game with intelligent agents in the Unity 3D game environment’, in 2020 International Conference on Electrical Engineering (ICEE), Sep. 2020, pp. 1–6, doi: 10.1109/ICEE49691.2020.9249828.

[26] D. Daylamani-Zad, L. B. Graham, and I. T. Paraskevopoulos, ‘Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games’, in 2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games), Sep. 2017, pp. 39–46, doi: 10.1109/VS-GAMES.2017.8055809.