Universitat Oberta de Catalunya

uoc.edu

Aplicació d'algoritmes d'intel·ligència artificial per al desenvolupament d'agents per als jocs de cartes "Brisca" i "Tute"

Rubén Arjonilla Zamora
Grau d'enginyeria informàtica
Àrea d'intel·ligència artificial

Nom Consultor/a: Joan M. Nuñez do Rio
Professor responsable de l'assignatura: Susana Acedo Nadal

Data Lliurament: 16/06/2024

FITXA DEL TREBALL FINAL

Títol del treball:	*Aplicació d'algoritmes d'intel·ligència artificial per a l'entrenament d'agents per als jocs de cartes "Brisca" i "Tute"*
Nom de l'autor:	*Rubén Arjonilla Zamora*
Nom del consultor/a:	*Joan M. Nuñez do Rio*
Nom del PRA:	*Susana Acedo Nadal*
Data de lliurament (mm/aaaa):	16/06/24
Titulació:	*Grau d'enginyeria informàtica*
Àrea del Treball Final:	*Intel·ligència artificial*
Idioma del treball:	Català
Nombre de crèdits:	12
Paraules clau	*aprenentatge computacional, Agent IA, cartes, brisca, tute*

Resum del Treball:

La Brisca i el Tute són dos jocs de cartes estratègics amb informació imperfecta que no compten amb estudis previs en l'àmbit de l'aprenentatge computacional, però que són prou interessants i presenten un especial interès en el camp de l'aprenentatge automàtic per la seva basant estratègica.

La investigació se centra en el desenvolupament d'agents capaços d'aprendre a jugar als dos jocs utilitzant tècniques d'aprenentatge supervisat (xarxes neuronals), aprenentatge genètic i aprenentatge per reforç, i abasta des de la implementació de l'entorn i la interfície gràfica d'usuari fins a l'entrenament dels models utilitzant diferents enfocaments per a cada tècnica.

Es detallen els models i resultats obtinguts junt amb una anàlisi qualitativa sobre la capacitat d'aprenentatge de cada tècnica i del rendiment dels models obtinguts per jugar als jocs. Es discuteixen les fortaleses i debilitats de les diferents tècniques i enfocaments, es detallen les limitacions trobades durant l'entrenament, i s'especifiquen possibles àrees de millora i futures vies d'investigació per a continuar avançant en l'estudi de l'aprenentatge computacional aplicat a jocs de cartes estratègics amb informació imperfecta.

Al final de la investigació, jugadors experimentats s'han enfrontat als diferents models obtinguts i han guanyat gairebé la totalitat de les partides contra els agents entrenats per reforç i el genètic, però ha estat més igualat quan s'han enfrontat als models supervisats. Els models obtinguts encara tenen molt marge de millora i queda molt per investigar en aquest camp.

Abstract:

Brisca and Tute are two strategic card games with imperfect information that have not been studied in the computational learning field, but they are quite interesting and have special interest in machine learning field for their strategic base.

This research focuses on the development of agents capable of learning to play both games using supervised learning techniques (neural networks), genetic algorithms and reinforcement learning, and it covers from the environment and graphical user interface development to the training of models using different approaches for each technique.

Models and results are detailed along a qualitatvie analysis of the learning capability of each technique and the models' performance playing the games. The strenghts and weaknesses of each technique and approachment are discussed, the limitations found during the training are detailed, and possible areas of improvement and future research directions are identified to continue advancing in computational learning applied to strategic card games with imperfect information.

At the end of this research, experienced players have faced the different obtained models and won almost all matches against reinforcement and genetic trained agents, but it was more balanced when they faced supervised models. The obtained models still have a lot of room to improve and there is much more to investigate in this field.

Índex

Llista de figures

Llista de taules

1. Resum

La investigació que s'ha dut a terme està centrada en el desenvolupament de l'entorn del joc de la Brisca i el Tute, l'entrenament de models aplicant diverses tècniques com l'aprenentatge supervisat (xarxes neuronals), algoritme genètic i aprenentatge per reforç, i l'anàlisi i comparativa de les diferents tècniques utilitzades i dels models obtinguts.

En primer lloc, l'entorn del joc permet simular partides per als jocs de la Brisca i el Tute, i permet escollir quines regles opcionals es vol utilitzar. Aquest entorn s'emprarà per a la generació del conjunt inicial de dades de l'aprenentatge supervisat i serà utilitzat per la resta d'algoritmes per obtenir la informació necessària, interactuar amb l'entorn i aprendre a jugar.

A més a més, s'han utilitzat diferents enfocaments amb cada tècnica amb l'objectiu de millorar els resultats obtinguts i reduir al màxim els temps d'entrenament. S'ha detallat cadascun dels experiments realitzats fins a aconseguir els enfocaments finals utilitzats per l'entrenament dels models.

S'ha analitzat i comparat l'eficàcia de cadascuna de les tècniques tenint en compte factors com la rapidesa en l'entrenament, la mida dels models, l'ús de recursos necessaris per a l'entrenament i per al seu ús, i el rendiment mostrat pels models. Els resultats mostren que l'aprenentatge supervisat és el més ràpid per a l'entrenament mentre que l'algoritme per reforç i el genètic requereixen més temps, fins al punt que el genètic ha resultat inviable per a l'entrenament amb el temps del qual es disposava.

Pel que fa als models generats, els obtinguts mitjançant l'aprenentatge supervisat i genètic són els més lleugers, mentre que els de reforç són els més pesats i els que requereixen més recursos, tant per al seu entrenament com per al seu ús. Els models que millors resultats han donat han estat els supervisats, pel fet que són els únics amb els quals s'ha pogut finalitzar l'entrenament per complet.

En el cas dels algoritmes genètics, el temps necessari per a l'entrenament no ha permès entrenar un model amb més de 1.000 generacions (s'han necessitat cinc dies per completar aquest entrenament), mentre que per als de l'aprenentatge per reforç només s'han pogut entrenar amb 3.000.000 de partides (els recursos necessaris són tan elevats que aquest és el límit de partides amb les quals s'ha aconseguit finalitzar l'entrenament).

2. Introducció

2.1. Context i justificació del Treball

La Brisca i el Tute són jocs de cartes d'entre dos i quatre jugadors que es juguen amb una baralla espanyola. L'objectiu és guanyar més punts que els rivals emportant-se les cartes jugades a les diferents rondes de la partida, on cada jugador ha de triar una de les cartes que disposa a la seva mà, i sumar els punts associats a elles.

A l'hora de triar una carta, els jugadors han de tenir en compte factors com les cartes de triomf, que són d'un coll específic que tindrà més preferència que els altres al llarg de tota la partida, el valor de les cartes jugades i les regles aplicades específiques de cadascun dels jocs per intentar sumar més punts que els rivals i així guanyar la partida.

Compten amb molts aficionats arreu del món i, a causa de la seva basant estratègica, presenten un especial interès en el camp de l'aprenentatge automàtic. L'estudi de mètodes d'aprenentatge automàtic aplicat a jocs de cartes estratègics proporciona una plataforma d'investigació fascinant i presenta reptes computacionals importants que obren noves oportunitats per a la investigació i desenvolupament.

Són jocs que requereixen una combinació d'estratègia, tàctica i habilitats matemàtiques, que els converteix en uns problemes de presa de decisions complexos i en un entorn de prova ideal per als diferents mètodes d'aprenentatge. Els agents han de ser capaços d'aprendre a jugar i adaptar-se a les diferents situacions que se li poden presentar en qualsevol moment de la partida, presentant un repte computacional significatiu.

Les tècniques d'aprenentatge automàtic disposen de múltiples enfocaments per a la creació d'agents que poden fer prediccions sense ser programats per a tal fi. Per exemple hi trobem l'aprenentatge supervisat, on els models aprenen a partir d'exemples que contenen el resultat esperat; l'aprenentatge genètic, on els individus evolucionen cap a solucions òptimes a través de la selecció natural; i l'aprenentatge per reforç, on els models aprenen a través de la interacció amb l'entorn i de l'experiència acumulada.

Cada tècnica té els seus avantatges i desavantatges, i poden resultades més o menys adequades segons el tipus de problema a resoldre. La seva combinació permet explorar nous mètodes d'aprenentatge i comprendre millor la naturalesa de la presa de decisions en els jocs estratègics.

Es poden trobar molts exemples d'aquestes tècniques aplicades a jocs d'estratègia com els escacs (Deep Blue [6] i AlphaZero [8]), Go (AlphaGo [10]), Starcraft 2 (AlphaStar [11]) i Póker (Libratus [14] i Pluribus [15]), però la seva aplicació a jocs com la Brisca i el Tute és pràcticament inexistent. Per aquest motiu, aquesta investigació se centrarà en aquests jocs.

En aquest estudi es faran servir tècniques d'aprenentatge supervisat, aprenentatge per reforç i algoritmes genètics als jocs de la Brisca i el Tute. S'analitzaran diverses partides dels diferents agents per tal de comparar les estratègies que aquests han seguit, i, a més, el lector podrà generar els seus propis agents, que seran creats segons una sèrie de paràmetres que es podran ajustar molt fàcilment en qualsevol moment.

Com a mètode d'aprenentatge supervisat s'ha optat per les xarxes neuronals per la seva capacitat de modelar relacions complexes entre les dades d'entrada i la seva capacitat de generalització. Aquestes estan compostes per capes de nodes interconnectats inspirats en la connexió entre neurones del cervell humà.

Un altre dels mètodes aplicats és el dels algoritmes genètics, que es basen en una tècnica d'optimització inspirada en la teoria de l'evolució natural. Es partirà d'una generació de

poblacions inicial aleatòria que representaran les solucions candidates que seran avaluades i comparades en funció del seu rendiment.

S'aplicaran tècniques d'aprenentatge per reforç, que se centren en com els agents poden aprendre a prendre decisions mitjançant una retroalimentació positiva o negativa en funció de les seves accions.

La investigació finalitza amb una comparativa qualitativa entre les diferents tècniques emprades identificant les fortaleses i debilitats de cadascuna, tenint en compte factors com la rapidesa en l'aprenentatge, els recursos necessaris per a l'entrenament, la capacitat d'adaptació dels agents a diferents situacions i l'eficàcia en la presa de decisions.

2.2. Objectius del treball

Els objectius del treball són els pilars fonamentals que guien i orienten cada pas de la investigació i proporcionen una direcció clara i ben definida.

2.2.1. Objectius generals

- Aplicació d'algoritmes d'aprenentatge supervisat, per reforç i genètics: implementació i utilització de diferents algoritmes d'aprenentatge per a l'entrenament d'agents capaços de jugar a la Brisca i al Tute.
- Contribució al coneixement sobre jocs de cartes amb informació imperfecte i aprenentatge automàtic: es contribueix al coneixement de l'aplicació de tècniques d'aprenentatge automàtic a jocs de cartes i a l'optimització d'estratègies en entorns amb informació imperfecta.
- Estudi d'adaptació dels agents a variacions en les regles del joc aplicades: Implica l'estudi de l'adaptabilitat dels agents per a diferents regles de joc.

2.2.2. Objectius específics

- Cerca de literatura sobre projectes similars: se cerquen projectes d'aprenentatge computacional existents basats en els jocs de la Brisca i el Tute.
- Desenvolupament de l'entorn del joc: implica el disseny i implementació de la infraestructura del programari necessària per a la simulació dels jocs amb totes les seves regles, variacions i modalitats.
- Desenvolupament de l'entorn gràfic: Disseny i implementació d'una interfície gràfica d'usuari (GUI) que permeti visualitzar les simulacions dels agents i avaluar-los personalment.
- Generació de dades per a l'entrenament dels algoritmes d'aprenentatge supervisat: Preparació de conjunts de dades rellevants que puguin ser emprats per a l'entrenament dels agents en l'aprenentatge supervisat.
- Desenvolupament del marc de treball: creació d'un entorn de desenvolupament complet i flexible que faciliti el procés d'entrenament dels agents.
- Anàlisi i comparativa dels agents: S'avaluen i comparen els agents obtinguts mitjançant mètriques i anàlisi d'enfrontaments entre agents amb l'objectiu d'identificar les seves fortaleses i debilitats.
- Anàlisi de partides dels agents: Implica l'anàlisi de partides jugades per agents en cerca d'estratègies, patrons, encerts i errors en les seves jugades.
- Anàlisi i comparativa dels algoritmes: Es busca l'avaluació i comparació dels mètodes aplicats en funció del seu rendiment (rapidesa en l'aprenentatge, capacitat d'adaptació a diferents situacions, recursos necessaris, eficàcia en la presa de decisions...).

2.3. Enfocament i mètode seguit

La metodologia que se seguirà al llarg del desenvolupament del treball és una metodologia seqüencial en cascada, pel fet que el començament d'una depèn en força mesura de la finalització de la fase anterior.

La majoria de les fases estan ben diferenciades les unes amb les altres i totes tenen un propòsit i sentit per si mateixes. En un principi, un cop començada una fase no caldrà tornar a fases anteriors, però, sí que es pot donar el cas què es detecti alguna anomalia, error o situació no prevista que requereixi dur a terme modificacions en algun component d'una fase anterior.

2.3.1. Fase 1: Desenvolupament de l'entorn del joc

En aquesta primera fase es desenvoluparà l'entorn del joc que permetrà realitzar simulacions de partides dels jocs de la Brisca i el Tute.

Haurà de contenir tota la informació necessària per al desenvolupament de les partides i definir totes les regles, específiques de cadascun dels jocs i opcionals, necessàries, així com de les diferents modalitats de joc.

L'entorn del joc haurà d'emmagatzemar l'estat de la partida (tipus de joc que s'està simulant, el nombre de jugadors, les cartes que cadascun d'ells disposa a la seva mà, les cartes que resten per robar i la carta de triomf), les accions que l'agent o humà pot portar a cap segons l'estat de la partida, la puntuació dels jugadors a cada ronda i l'acumulada de tota la partida entre d'altres.

En aquesta primera versió, les accions dels jugadors seran aleatòries, però, sempre seguiran les normes especificades.

2.3.2. Fase 2: Desenvolupament de la interfície gràfica d'usuari (GUI)

La segona fase té el propòsit de desenvolupar una versió gràfica dels jocs que permetrà, per una banda, visualitzar les partides simulades pels agents (que d'entrada prendran decisions aleatòries) i poder fer-ne un seguiment, i, per l'altre, permetrà a un jugador humà jugar contra els diferents agents (només es preveu que hi pugui haver un jugador humà alhora).

2.3.3. Fase 3: Desenvolupament del marc de treball per a l'entrenament

En aquesta fase es desenvoluparà el marc de treball necessari per a l'entrenament dels agents i de les mètriques que permetran avaluar-los. El marc de treball haurà de permetre, d'una forma senzilla, la càrrega de les dades i dels diferents agents i l'entrenament dels agents per a cadascun dels algoritmes triats.

Aquesta fase també inclou l'estudi dels diferents algoritmes per tal de realitzar una implementació correcta de cadascun d'ells i serà un punt clau de cara a un entrenament òptim dels agents.

2.3.4. Fase 4: Entrenament dels agents

En finalitzar aquesta fase, s'hauran generat els diferents agents utilitzant el marc de treball desenvolupat prèviament. Caldrà dur a terme múltiples experiments i entrenaments mentre es van ajustant els diferents paràmetres de cadascun dels algoritmes, i s'hauran d'emprar les mètriques adequades per a avaluar-los i poder escollir els millors en cadascuna de les modalitats i regles dels jocs.

Es generarà un model amb cadascun dels algoritmes i per a cadascuna de les modalitats i variacions per tal de trobar un agent optimitzat a cada cas (per exemple, un agent optimitzat per a jugar a la Brisca amb dos jugadors no serà òptim per a jugar a la versió per a quatre jugadors en la modalitat de partida en equips).

2.3.5. Fase 5: Comparativa i anàlisi dels agents obtinguts i algoritmes emprats

En aquesta darrera fase de desenvolupament, es durà a terme, en primera instància, la comparativa entre els diferents agents obtinguts amb l'aplicació dels diferents algoritmes per a cada modalitat de joc (no es compararan 2 agents de modalitats diferents), i, en segona instància, s'analitzaran els diferents algoritmes emprats per tal de trobar-ne les fortaleses i debilitats per a l'entrenament general dels agents.

2.3.6. Fase 6: Redacció de la memòria

Durant aquesta fase es realitzarà la redacció de la memòria del treball, que és l'única que es podrà dur a terme en paral·lel a la resta de fases a mesura que es va avançant en la investigació.

2.3.7. Fase 7: Presentació del treball i defensa pública

La darrera fase del treball es basa en la presentació del treball i la defensa pública, i tindrà lloc una vegada finalitzada la investigació.

2.3.8. Prestacions de l'ordinador

A continuació es detallen les prestacions de l'equip en el qual es duran a terme els entrenaments dels agents.

- Processador: Intel Core i7-7700K CPU @ 4.20GHz
- Memòria RAM: Kingston FURY Beast DDR4 3600 MHz 64GB 2x32GB CL18
- Targeta gràfica: MSI GeForce GTX 1070 Gaming 8 GB GDDR5
- Placa mare: MSI Z270 PC MATE

2.4. Planificació del treball

La planificació del treball és un procés essencial per al desenvolupament de la investigació, ja que estableix el full de ruta a seguir per a assolir els objectius proposats de manera eficient i efectiva.

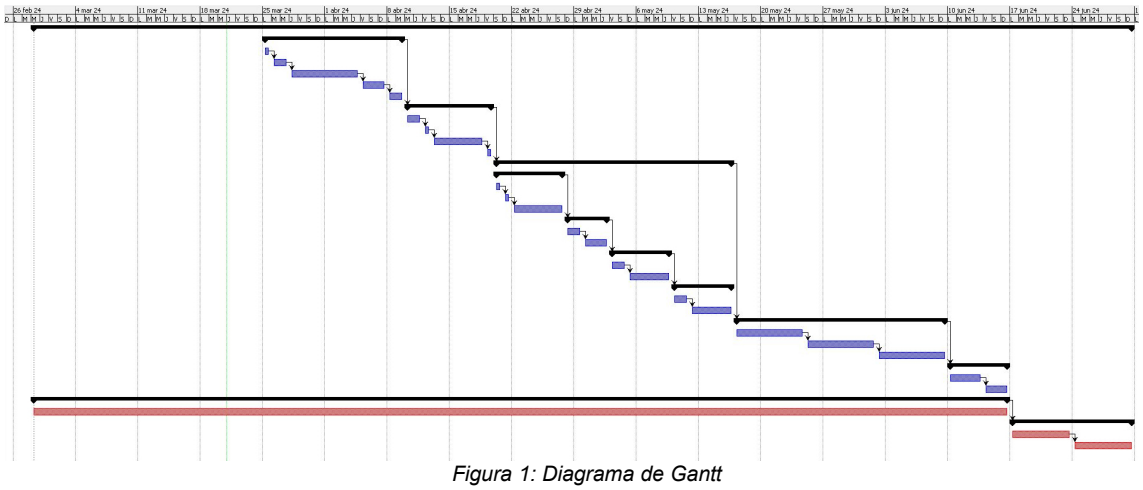

Figura 1: Diagrama de Gantt

Planificació del TFG	**133 days**	**28/02/24 8:00**	**30/06/24 17:00**
Fase 1: Desenvolupament de l'entorn del joc	**16 days**	**25/03/24 8:00**	**9/04/24 17:00**
Anàlisi dels jocs i detecció dels elements necessaris	1 day	25/03/24 8:00	25/03/24 17:00
Disseny de les classes	2 days	26/03/24 8:00	27/03/24 17:00
Implementació bàsica dels jocs	8 days	28/03/24 8:00	4/04/24 17:00
Implementació de les variacions	3 days	5/04/24 8:00	7/04/24 17:00
Simulacions aleatòries i validació de l'entorn	2 days	8/04/24 8:00	9/04/24 17:00
Fase 2: Desenvolupament de la interfície gràfica d'usuari (GUI)	**10 days**	**10/04/24 8:00**	**19/04/24 17:00**
Investigació de llibreries per al desenvolupament de la GUI	2 days	10/04/24 8:00	11/04/24 17:00
Disseny de la GUI	1 day	12/04/24 8:00	12/04/24 17:00
Implementació de la GUI	6 days	13/04/24 8:00	18/04/24 17:00
Validació de la GUI	1 day	19/04/24 8:00	19/04/24 17:00
Fase 3: Desenvolupament del marc de treball per a l'entrenament	**27 days**	**20/04/24 8:00**	**16/05/24 17:00**
General	**8 days**	**20/04/24 8:00**	**27/04/24 17:00**
Definició dels estats per a l'entrenament	1 day	20/04/24 8:00	20/04/24 17:00
Implementació dels estats	1 day	21/04/24 8:00	21/04/24 17:00
Desenvolupament del marc de treball per a l'entrenament	6 days	22/04/24 8:00	27/04/24 17:00
Aprenentatge supervisat	**5 days**	**28/04/24 8:00**	**2/05/24 17:00**
Recerca d'informació sobre l'algoritme i la seva implementació	2 days	28/04/24 8:00	29/04/24 17:00
Desenvolupament de l'algoritme	3 days	30/04/24 8:00	2/05/24 17:00
Algoritme genètic	**7 days**	**3/05/24 8:00**	**9/05/24 17:00**
Recerca d'informació sobre l'algoritme i la seva implementació	2 days	3/05/24 8:00	4/05/24 17:00
Desenvolupament de l'algoritme	5 days	5/05/24 8:00	9/05/24 17:00
Aprenentatge per reforç	**7 days**	**10/05/24 8:00**	**16/05/24 17:00**
Recerca d'informació sobre l'algoritme i la seva implementació	2 days	10/05/24 8:00	11/05/24 17:00
Desenvolupament de l'algoritme	5 days	12/05/24 8:00	16/05/24 17:00
Fase 4: Entrenament dels agents	**24 days**	**17/05/24 8:00**	**9/06/24 17:00**
Entrenament i ajustament dels paràmetres - Aprenentatge Supervisat	8 days	17/05/24 8:00	24/05/24 17:00
Entrenament i ajustament dels paràmetres - Algoritmes genètics	8 days	25/05/24 8:00	1/06/24 17:00
Entrenament i ajustament dels paràmetres - Aprenentatge per reforç	8 days	2/06/24 8:00	9/06/24 17:00
Fase 5: Comparativa i anàlisi dels agents obtinguts i algoritmes emprats	**7 days**	**10/06/24 8:00**	**16/06/24 17:00**
Anàlisi i comparativa dels agents	4 days	10/06/24 8:00	13/06/24 17:00
Anàlisi i comparativa dels algoritmes	3 days	14/06/24 8:00	16/06/24 17:00
Fase 6: Redacció de la memòria	**119 days**	**28/02/24 8:00**	**16/06/24 17:00**
Redacció i revisió de la memòria	119 days	28/02/24 8:00	16/06/24 17:00
Fase 7: Presentació del treball i defensa pública	**14 days**	**17/06/24 8:00**	**30/06/24 17:00**
Preparació de la presentació	7 days	17/06/24 8:00	23/06/24 17:00
Defensa del treball	7 days	24/06/24 8:00	30/06/24 17:00

Figura 2: Desglossament de les fases

2.4.1. Avaluació de riscos

La investigació no està absenta de riscos que representen esdeveniments o situacions incertes que poden impactar negativament en l'assoliment dels objectius o el compliment dels terminis d'entrega.

Codi	R01	**Nom**	Subestimació de la durada de les tasques		
Causa	Hi ha tasques que requereixen més temps del previst.				
Descripció	S'ha subestimat alguna de les tasques que ha provocat un retard a les fases posteriors.				
Conseqüència	En tractar-se d'una metodologia en cascada, un retard en una de les tasques provoca un retard a les tasques posteriors, i, per tant, un retard en la finalització del projecte.				
Probabilitat	Mitja	**Impacte**	Mitjà	**Nivell**	Mitjà

Codi	R02	**Nom**	Imprevistos dins del projecte		
Causa	Complicacions, incompatibilitat de llibreries...				
Descripció	Durant el desenvolupament del projecte han sorgit complicacions o incompatibilitats entre les versions de les llibreries utilitzades fins al moment.				
Conseqüència	Es poden produir retards en l'execució projecte que poden repercutir en les tasques posteriors o, fins i tot, poden fer replantejar o eliminar algun dels objectius inicials.				
Probabilitat	Baixa	**Impacte**	Alt	**Nivell**	Mitjà

Codi	R03	**Nom**	Imprevistos personals		
Causa	Malaltia, viatges, feina.				
Descripció	Durant el transcurs del treball poden sorgir imprevistos personals com poden ser malalties (personal o familiars), viatges no planificats o, inclús, un excés de feina que requereixi treballar hores addicionals.				
Conseqüència	Es disposarà de menys temps del que s'havia previst per a la realització de les tasques, fet que pot provocar retards a les tasques o simplificacions que poden afectar el resultat final.				
Probabilitat	Baixa	**Impacte**	Mitjà	**Nivell**	Mitjà

Codi	R04	**Nom**	Temps d'entrenament elevat		
Causa	Ineficiència de l'entorn del joc o baixes prestacions de l'equip on es realitza l'entrenament.				
Descripció	El temps de simulació i d'entrenament dels agents és molt elevat.				
Conseqüència	Es duen a terme menys experiments i menys ajustaments de paràmetres que poden resultar en agents menys optimitzats.				
Probabilitat	Baix	**Impacte**	Alt	**Nivell**	Mitjà

Taula 1: Avaluació de riscos

2.4.2. Accions de mitigació

La gestió efectiva del temps i els riscos és fonamental per a garantir l'èxit del projecte. Per tant, s'abordaran estratègies específiques correctores i de mitigació per afrontar els riscos identificats i mantenir el progrés cap als objectius establerts, sempre tenint en compte la data límit de l'entrega del projecte.

Codi	**Acció**	**Tipus**	**Risc residual**
A1R01	Se segueix amb la planificació i es redueix el temps per a l'entrenament dels agents.	Mitigadora	Baix
A2R01	Es redueix el nombre d'algoritmes per a l'entrenament.	Mitigadora	Baix
A1R02	Se segueix amb la planificació i es redueix el temps per a l'entrenament dels agents.	Mitigadora	Baix
A2R02	Es redueix el nombre d'algoritmes per a l'entrenament.	Mitigadora	Baix
A1R03	Se segueix amb la planificació i es redueix el temps per a l'entrenament dels agents.	Mitigadora	Baix
A2R03	Es redueix el nombre d'algoritmes per a l'entrenament.	Mitigadora	Baix
A1R04	Es planteja el lloguer d'un servidor al núvol, o, fins i tot, la compra d'una nova computadora.	Correctora	Mitjà
A2R04	S'adapta l'entorn de joc per permetre l'execució en paral·lel de les simulacions.	Correctora	Baix
A3R04	S'accepta que es produiran agents no òptims.	Acceptadora	Mitjà

Taula 2: Accions de mitigació per als riscos identificats

2.5. Breu sumari de contribucions i productes obtinguts

Productes obtinguts:

- Entorn dels jocs que permet aplicar tècniques d'aprenentatge computacional.
- Entorn visual del joc que permet avaluar els models obtinguts (tant agent vs. agent com humà vs. agent).
- Un agent supervisat amb dues capes de neurones per a cada joc (Brisca, Tute i Tute amb només l'obligació d'assistir) per a les modalitats de dos, tres, quatre jugadors i per equips (12 agents) que són capaços de jugar aplicant qualsevol combinació de regles opcionals.
- Un agent supervisat amb tres capes de neurones per a cada joc (Brisca, Tute i Tute amb només l'obligació d'assistir) per a les modalitats de dos, tres, quatre jugadors i per equips (12 agents) que són capaços de jugar aplicant qualsevol combinació de regles opcionals.
- Un agent entrenat genèticament per a la modalitat de dos jugadors de la Brisca que pot jugar sense aplicar cap regla opcional.
- Un agent entrenat per reforç per a cada joc (Brisca, Tute i Tute amb només l'obligació d'assistir) per a les modalitats de dos, tres, quatre jugadors i per equips (12 agents) que pot jugar sense aplicar cap regla opcional.

Contribucions:

- Contribució en l'avenç del coneixement en l'àmbit de la intel·ligència artificial aplicada a jocs de cartes.
- Oferir noves perspectives i oportunitats per a investigacions futures.

2.6. Breu descripció dels capítols de la memòria

En el capítol “Estat de l'art” es du a terme una anàlisi de l'estat de l'art l'aplicació d'intel·ligència artificial a jocs d'estratègia i cartes. Es revisen investigacions prèvies, algoritmes emprats i resultats obtinguts en l'àmbit de la Brisca i el Tute.

El capítol “La Brisca i el Tute” es dedica a la descripció detallada d'aquests jocs, incloent-hi les seves regles i possibles variacions. S'estableix una base per comprendre els reptes que plantegen aquests jocs en l'aplicació de tècniques d'intel·ligència artificial per al seu aprenentatge.

En el capítol “Conceptes tècnics” es presenten les característiques de l'equip utilitzat en els experiments i una idea general del funcionament dels algoritmes escollits.

En el capítol “Metodologia” es presenta la metodologia utilitzada en el desenvolupament de la investigació. S'inclou la implementació de l'entorn del joc, així com el disseny i implementació dels algoritmes basats en xarxes neuronals, algoritmes genètics i algoritmes per reforç. Es defineixen els estats del joc més rellevants i els experiments realitzats per a cada algoritme.

En el capítol “Resultats” es mostren els resultats assolits durant el desenvolupament i es presenten diferents gràfiques que comparen el rendiment dels agents a partir dels seus emparellaments en simulacions de partides.

En el capítol “Discussió” s'analitzen els resultats aconseguits i es discuteixen les fortaleses i debilitats de les tècniques i enfocaments seguits, així com les limitacions de les diferents tècniques.

En el capítol “Conclusions” es presenten les conclusions de la investigació, destacant les lliçons apreses i assenyalant direccions futures per a la investigació en aquest camp.

3. Estat de l'art

La intel·ligència artificial (en anglès, Artificial Intelligence - AI) està enfocada al desenvolupament de sistemes i programes, anomenats agents, que poden executar tasques on es sol requerir intel·ligència humana. El reconeixement de patrons, la presa de decisions, el processament del llenguatge natural, la resolució de problemes complexos o la interacció amb l'entorn en són exemples.

És aplicada en una àmplia varietat de camps i sectors com la medicina, la indústria automotriu, l'atenció al client, la seguretat, la robòtica, els videojocs i les finances entre molts d'altres, i cada vegada estan més presents en el nostre dia a dia, des dels assistents virtuals en els nostres telèfons intel·ligents fins als sistemes de recomanació en plataformes de streaming o les tecnologies de conducció autònoma en vehicles.

3.1. Història

El concepte de màquines que poden aprendre i adaptar-se per si mateixes es remunta als primers dies de la informàtica. Alan Turing, un dels pares de la ciència de la computació, va proposar la idea que les màquines podrien pensar a través de l'experiència en el seu article "Computing Machinery and Intelligence" de 1950. Però, no va ser fins dècades més tard quan el terme “aprenentatge computacional” (en anglès, Machine Learning) es va popularitzar. [1]

Una de les primeres fites en el desenvolupament de l'aprenentatge computacional va ser el perceptró, suggerit per Frank Rosenblatt el 1958, una de les primeres xarxes neuronals artificials que tenia la capacitat d'aprendre a reconèixer patrons simples en les dades, la qual disposava únicament d'una capa de neurones. Però les limitacions de la tecnologia de l'època i la manca de grans conjunts de dades (en anglès, Big Data) per al seu entrenament van fer disminuir l'interès d'aquest camp. [2, 3]

A la dècada dels vuitanta, aquest interès va tornar a augmentar gràcies al desenvolupament de noves tècniques com els algoritmes d'aprenentatge automàtic basats en arbres de decisió, les xarxes neuronals que contenien, a diferència del perceptró, diverses capes intermèdies i el desenvolupament de l'algoritme Q-Learning per a l'aprenentatge per reforç (en anglès, Reinforcement Learning - RL). Aquestes noves tècniques, junt amb els avenços en els camps de l'optimització i l'estadística computacional va permetre als investigadors tractar problemes d'aprenentatge més complexos. [3, 4]

Al 1992, J.H. Holland va proposar els algoritmes genètics (en anglès, Genetic Algorithms – GA), que es basen en la teoria de la selecció natural de Charles Darwin. Aquests algoritmes simulaven els processos de selecció natural i evolució biològica en sistemes informàtics. [5]

Però, no ha estat fins als últims anys en què l'aprenentatge profund (en anglès, Deep Learning – DL) ha permès als agents aprendre representacions molt complexes.

3.2. IA Aplicada a jocs d'estratègia i cartes

Els jocs estratègics són un entorn de prova perfecte per a l'experimentació i millora dels algoritmes d'aprenentatge automàtic per la seva complexitat, incertesa i interacció entre diferents jugadors i han exercit un paper crucial en el desenvolupament d'agents capaços de resoldre problemes complexos.

Jocs com els escacs, Go, Starcraft II i Pòquer presenten una sèrie de reptes que requereixen una combinació d'habilitats cognitives, tàctiques i estratègiques. El fet que les IA actuals puguin analitzar grans quantitats de dades, identificar patrons complexos i aprendre a partir de l'experiència acumulada ha estat fonamental per afrontar-los.

En el cas dels escacs, la intel·ligència artificial ha estat utilitzada des de la dècada dels noranta per a desenvolupar agents capaços de competir contra els millors jugadors humans del món. Una de les fites més significatives va ser la victòria de l'agent “Deep Blue" sobre el campió mundial Garry Kasparov el 1997. Aquest agent, desenvolupat per IBM, es basa en l'algoritme MiniMax amb poda alfa-beta i avalua les posicions del tauler per escollir la millor jugada en cada situació. [2, 6]

Des de llavors, s'han dut a terme la creació d'agents molt més sofisticats com poden ser Stockfish, un motor d'escacs de codi obert que es basa en la cerca heurística i l'optimització per avaluar les posicions d'una posició concreta del tauler, i AlphaZero, una xarxa neuronal profunda, desenvolupada per DeepMind de Google, que utilitza tant tècniques d'aprenentatge profund com d'aprenentatge per reforç. [7, 8]

La intel·ligència artificial es va enfrontar a un repte encara més gran amb el joc Go a causa de la seva alta complexitat pel gran nombre de possibles estats que aquest pot tenir i a l'absència d'una funció d'avaluació clara, pel fet que no es pot definir un valor específic a les peces.

No obstant això, l'agent AlphaGo, també desenvolupat per DeepMind, va aconseguir la victòria contra un professional contra el tres vegades campió Europeu, Fan Hui, el 2015, al segon millor jugador del món, Lee Se-Dol, el 2016 i al millor, Ke Jie, el 2017, demostrant la gran capacitat de l'aprenentatge computacional per a dominar jocs estratègics complexos. Anàlogament a l'agent AlphaZero, AlphaGo també utilitza una xarxa neuronal profunda que utilitza tècniques d'aprenentatge profund i d'aprenentatge per reforç. [9, 10]

Tots els jocs anteriors entren dins la categoria del que s'anomena “jocs d'informació perfecta”, on cada jugador coneix l'estat exacte del joc en qualsevol moment de la partida. Per contra, en els jocs d'informació imperfecta com els que es descriuran a continuació, els jugadors tenen coneixements parcials de l'estat del joc en un moment donat, aquesta informació sol fer referència a la pròpia del jugador.

Continuant amb els jocs d'estratègia, DeepMind va dissenyar una intel·ligència artificial, anomenada AlphaStar, que va aprendre a jugar al videojoc d'estratègia en temps real StarCraft II. En aquest joc es desconeix per complet les accions que el rival està realitzant i només s'aconsegueix part de la informació si una de les unitats del jugador entra en contacte amb una altra unitat del rival. Aquest agent va guanyar a diversos jugadors humans professionals en partides oficials el 2019, convertint-se en una fita significativa en la investigació de la intel·ligència artificial.

AlphaStar va ser dissenyat utilitzant un enfocament combinat d'aprenentatge supervisat de xarxes neuronals profundes i aprenentatge per reforç, igual que els seus anàlegs AlphaZero i AlphaGo, i va ser entrenada utilitzant una infraestructura de computació distribuïda i paral·lela, que li va permetre accelerar el procés d'entrenament en poder realitzar múltiples simulacions alhora. [11]

Ja en el món dels jocs de cartes, concretament el Pòquer Texas hold'em, trobem projectes com Libratus i Pluribus, dos models d'intel·ligència artificial on s'han aplicat diferents tècniques d'aprenentatge automàtic i que representen avenços significatius en aquest àmbit. Ambdós han mostrat molt bons resultats, hi han aconseguit guanyar a jugadors professionals en aquesta modalitat de Pòquer. [27, 28]

El projecte Libratus va ser desenvolupat per investigadors de la universitat de Carnegie Mellon, EUA, va ser presentat el 2017. Aquest projecte utilitza una combinació de tècniques d'aprenentatge automàtic segons l'estat del joc, del qual en crea abstraccions per tal de minimitzar els possibles estats. [14]

Per la seva part, el projecte Pluribus, també desenvolupat per investigadors de Carnegie Mellon, va ser presentat el 2019. Va ser entrenat amb una combinació d'algoritmes de cerca i xarxes neuronals utilitzant mètodes d'aprenentatge per reforç, on el model aprenia de la retroalimentació de les partides disputades contra versions anteriors del mateix model i de partides contra jugadors reals. [15]

3.3. Agents intel·ligents jugadors de la Brisca i el Tute

S'han trobat dos projectes desenvolupats en Python, “PyBrisca” i “Briscas”, que implementen l'entorn del joc de la Brisca i l'entrenament de diversos agents.

“PyBrisca” inclou una implementació gràfica del joc i permet jugar contra altres persones reals, contra un model que tria accions aleatòries i contra un model basat en regles que fa servir probabilitats per a la tria de la millor jugada. Aquest projecte empra la realitat augmentada, utilitzant cartes reals, perquè el model pugui “veure” les cartes en joc i decidir quina és la millor acció. Aquest projecte permet simular partides fins a 4 jugadors. [18]

Per la seva banda, “Briscas” és una implementació de l'entorn del joc sense interfície gràfica, i està pensat per a simular partides entre els diferents agents. En aquest cas, s'han implementat els models “random”, on l'agent du a terme accions aleatòries, “local player”, on l'agent tria la carta més dèbil que li permet guanyar la ronda, “smart player”, on l'agent és similar a l'anterior, però afegeix probabilitats i “KNN player”, un model supervisat entrenat amb el mètode “K-Nearest Neighbours”. El projecte preveia generar altres models aplicant mètodes de ML, però no es van acabar duent a terme. Com a punt negatiu, aquest projecte només preveu 2 jugadors. [19]

Tot i que són interessants, la manca d'investigació en les tècniques d'aprenentatge previstes en aquesta investigació reforça la necessitat d'omplir el forat i presenta una oportunitat per l'exploració de nous enfocaments i estratègies.

Aquesta investigació permetrà avançar en l'aprenentatge automàtic aplicat a jocs de cartes estratègics. La investigació busca desenvolupar una solució per als jocs de la Brisca i el Tute, i, al mateix temps, obrir nous camps d'investigació en el camp de la IA i aquesta mena de jocs.

4. La Brisca i el Tute

4.1. La Brisca

La brisca [12] és un joc de cartes que té els seus orígens a l'àrea del Mediterrani i que està molt estès als països hispanoparlants de l'Amèrica Llatina. Té unes regles bàsiques que la majoria de les regions segueixen, però n'hi ha d'altres que poden variar segons el país, i, fins i tot, pot variar segons la tradició familiar.

Les regles que es descriuran i seguiran al llarg de tot el treball són aquelles amb les quals vaig aprendre a jugar durant la meva infància, però, també es tindran en compte aquelles que s'han considerat interessants i que poden donar pas a altres estratègies i tàctiques a seguir, que seran definides com a opcionals i seran decidides abans de començar la partida.

4.1.1. Regles bàsiques

El joc permet de 2 a 4 jugadors i es juga amb una baralla Espanyola sense incloure els 8 i els 9. En el cas de 4 jugadors existeixen dues modalitats, tots contra tots i per equips.

Cada carta té associat un valor que indica el total de punts que sumarà el jugador que les guanyi, a més, el valor de les cartes indicarà la preferència entre les unes i les altres. La taula 3 mostra el valor de cada carta.

Carta	As	3	Rei	Cavall	Sota	7	6	5	4	2
Valor	11	10	4	3	2	0	0	0	0	0

Taula 3: Valor de les cartes al joc de la Brisca

Les cartes sense valor també tenen preferència les unes amb les altres. En aquest cas, la preferència es calcula segons el número de carta, sempre de major a menor.

Per tant, l'ordre de preferència de les cartes és:
As > 3 > Rei > Cavall > Sota > 7 > 6 > 5 > 4 > 2.

El valor total de la baralla és, en conseqüència, 120 punts, tal com es pot veure a la figura 3.

$$4 \cdot 11 + 4 \cdot 10 + 4 \cdot 4 + 4 \cdot 3 + 4 \cdot 2 = 44 + 40 + 16 + 12 + 8 = 120$$

Figura 3: Valor total de les cartes al joc de la Brisca

A l'inici de la partida hi haurà una carta, anomenada triomf, el qual el seu coll, anomenat el coll del triomf, tindrà preferència sobre la resta de cartes dels altres colls. És a dir, si per exemple, el triomf és el 2 d'ors, qualsevol carta d'aquest coll tindrà preferència sobre la resta de colls, indiferentment del seu valor. Les cartes del coll del triomf segueixen la preferència normal descrita anteriorment.

Exemples on el triomf és qualsevol carta del coll d'ors:

1. El 2 d'ors té preferència sobre l'As de bastos.
2. El 7 d'ors té preferència sobre el 2 d'ors.

Per altra banda, el coll de ronda, que correspon al coll de la primera carta jugada a cada ronda, tindrà preferència sobre la resta de colls, a excepció del coll del triomf.

Exemples on el triomf és qualsevol carta del coll d'ors i la primera carta de la ronda és l'As de bastos:

1. El 2 d'ors té preferència sobre l'As de bastos.
2. L'As de bastos té preferència sobre el 3 de bastos.
3. L'As de bastos té preferència sobre l'As d'espases.

4.1.2. Com es juga

Abans de començar a jugar, es designarà, de forma aleatòria, al repartidor inicial de les cartes, el qual canviarà en finalitzar cadascuna de les partides (se seguirà l'ordre en el sentit de les agulles del rellotge).

Es repartiran tres cartes a cada jugador, de forma saltejada i en sentit horari, i, finalment, s'extraurà una última carta, el triomf, que serà visible per a tots els jugadors i marcarà el coll que tindrà més preferència durant tota la partida. La resta de cartes quedaran cap per vall i seran robades pels jugadors un cop finalitzada una ronda.

A cada ronda, tots els jugadors han de jugar una de les tres cartes que tenen a la seva mà. Començarà a jugar la primera carta el jugador situat a l'esquerra del repartidor, i la resta de jugadors jugaran la seva carta en sentit horari i sense cap mena de restricció (poden jugar la carta que vulguin segons els seus interessos).

La ronda finalitza una vegada tots els jugadors han jugat la seva carta. En aquest moment, es comprovarà qui ha utilitzat la carta amb més preferència seguint l'ordre de preferència descrit anteriorment, i sumarà, a la seva puntuació de la partida, el valor de les cartes jugades d'aquesta ronda.

El guanyador de la ronda serà aquell jugador que hagi jugat la carta amb més preferència del mateix coll de ronda si no s'ha jugat cap carta del coll del triomf. En canvi, si s'ha jugat, com a mínim, una carta del coll del triomf, el guanyador serà el jugador que hagi jugat la carta amb més preferència d'aquest coll.

A més, no és necessari realitzar el recompte un cop finalitzada la ronda. Cada jugador haurà de portar un recompte mental de la puntuació de la resta de jugadors per tal de jugar estratègicament al llarg de la partida.

Els jugadors robaran una carta de la baralla cada vegada que finalitza una ronda, començant pel guanyador de l'anterior i seguint el sentit horari, i s'iniciarà una nova ronda, on serà el guanyador de l'anterior qui començarà jugant una carta de la seva mà.

Es continuaran jugant rondes seguint el procediment descrit fins que s'esgotin totes les cartes de la baralla i els jugadors ja no puguin robar més cartes.

En el cas de 2 i 4 jugadors, un dels jugadors haurà de robar la carta de triomf un cop s'hagin esgotat totes les cartes de la baralla. Aquest fet és un avantatge i un desavantatge alhora, pel fet que la resta de jugadors seran coneixedors d'una de les cartes de la seva mà, però, alhora, aquesta sempre serà una carta del coll prioritari de la partida.

En el cas de 3 jugadors, la carta de triomf no serà robada per cap dels jugadors i quedarà sempre a la vista. Aquesta carta només podrà ser intercanviada si es juga amb una de les regles opcionals que es descriu en un apartat posterior.

Quan ja no quedin cartes a la baralla, es jugaran les tres últimes rondes seguint el procediment descrit, amb l'excepció que, ara, ja no es podran robar més cartes i els jugadors es quedaran sense cartes. En aquest moment, es realitzarà el recompte de la puntuació final de cada jugador i aquell que tingui la puntuació més alta serà el guanyador de la partida.

En cas d'empat entre dos o més jugadors amb puntuació màxima, es considerarà que tots aquests han guanyat la partida.

Per acabar, en la modalitat de quatre jugadors en equips, el joc segueix les mateixes regles descrites anteriorment i els equips estaran formats per jugadors intercalats. És a dir, si tenim els jugadors “1”, “2”, “3” i “4” en sentit horari, l'equip 1 estarà format pels jugadors “1” i “3”, i l'equip 2 pels jugadors “2” i “4”. Aquests equips es definiran al principi de la partida i quedaran inalterats durant el seu transcurs.

L'única diferència amb la modalitat normal és que, ara, la puntuació final d'un equip serà la suma de la puntuació dels seus integrants.

4.2. El Tute

El Tute [13] és un joc de cartes que té els seus orígens a Espanya i que està molt estès als països hispanoparlants de l'Amèrica Llatina, així com en alguns països de l'est i Itàlia. De manera anàloga a la Brisca, es juga amb totes les cartes de la baralla Espanyola sense incloure els 8 i els 9.

Les regles que es descriuran i seguiran al llarg de tot el treball són aquelles amb les quals vaig aprendre a jugar durant la meva infància, però, també es tindran en compte aquelles que s'han considerat interessants i que poden donar pas a altres estratègies i tàctiques a seguir, que seran definides com a opcionals i que seran decidides abans de començar la partida.

4.2.1. Regles bàsiques

Tant el nombre de jugadors, les modalitats, el valor de les cartes i les preferències d'aquestes sobre la resta de cartes i colls segueixen el mateix criteri que al joc de la Brisca. Per tant, les regles bàsiques explicades anteriorment també tenen validesa per al Tute.

Els jugadors tindran 8 cartes a la mà en comptes de les 3 que es tenen a la Brisca (tot i que hi ha regions on es reparteixen totes les cartes entre tots els jugadors).

Existeixen unes restriccions a l'hora de jugar una carta de la mà que tots els jugadors, a excepció del jugador inicial de la ronda, hauran de seguir: assistir, muntar, fallar, trepitjar i contrafallar:

- Assistir: El jugador haurà de jugar una carta del coll de ronda en cas que en disposi d'alguna.
- Muntar: A més, si pot, la carta jugada haurà de tenir més preferència que la carta amb més preferència de la ronda, llevat que s'hagi jugat una carta del coll del triomf, cas en què només haurà d'assistir.
- Fallar: Si el jugador no pot assistir, haurà de jugar una carta del coll del triomf.
- Trepitjar: En cas de fallar, si ja hi ha jugada una carta del coll del triomf (algun altre jugador ha fallat anteriorment), el jugador haurà de jugar una carta de triomf amb major preferència que la carta de triomf amb més preferència de la ronda. En cas que el jugador no disposi d'una carta del coll del triomf amb major preferència, no estarà obligat a fallar i podrà contrafallar.
- Contrafallar: Si el jugador no pot ni assistir ni fallar, podrà jugar la carta que desitgi sense cap restricció.

4.2.2. Càntics i Tute de cavalls o reis

Un càntic es dona quan un jugador disposa del rei i el cavall d'un mateix coll a la seva mà. Es diu que “canta 20 en X”, essent X el coll del rei i el cavall dels quals disposa, quan el coll és diferent del coll del triomf, i que “canta les 40” quan són del coll del triomf.

Quan un jugador disposa d'un càntic, ha d'indicar-ho a la resta de jugadors, i, si aconsegueix guanyar una ronda mentre encara disposa del rei i el cavall a la seva mà, sumarà 20 punts addicionals a la seva puntuació final en cas que hagi “cantat les 20”, i 40 si ha “cantat les 40”.

Un cop un jugador ha finalitzat un càntic, no podrà tornar a realitzar un càntic sobre aquest mateix coll, però sí sobre la resta.

Els càntics s'han d'indicar abans d'iniciar una ronda, i només està permès un càntic per jugador i ronda. Si un mateix jugador disposa de diversos càntics en una mateixa ronda, haurà d'escollir sobre quin dels colls vol dur a terme el càntic, però, en cas que un dels càntics sigui “les 40”, aquest sempre tindrà preferència sobre l'altre i no podrà triar.

Per altra banda, quan un jugador aconsegueix reunir els quatre cavalls o els quatre reis a la seva mà (jugada anomenada “Tute”), ho indicarà a la resta de jugadors, i, immediatament, guanyarà la partida amb la puntuació màxima, independentment de la puntuació de cada jugador en aquell moment. A més, la resta de jugadors no sumaran cap punt en aquella partida.

4.3. Regles opcionals

A continuació es descriuran aquelles regles que són opcionals (no són aplicades a totes les regions) i que s'ha cregut que poden donar pas a estratègies i tàctiques a seguir diferents segons si són aplicades o no.

4.3.1. Intercanvi de la carta de triomf

Tot i que es tracta d'una regla que se sol seguir a la majoria de les regions, no tothom l'aplica, per aquest motiu s'ha definit com a opcional.

L'activació d'aquesta regla permet a un jugador intercanviar la carta de triomf per una carta de la seva mà durant el seu torn si es donen unes condicions específiques.

Si la carta de triomf és una carta amb valor (As, 3, Rei, Cavall o Sota) i un jugador té a la seva mà el 7 del mateix coll del triomf, podrà intercanviar, si així ho vol, el 7 amb la carta de triomf. En canvi, si la carta de triomf no té valor (7, 6, 5 o 4) i un jugador té a la seva mà el 2 del mateix coll del triomf, podrà intercanviar, si així ho vol, el 2 amb la carta de triomf durant el seu torn.

A més, un jugador no està restringit a un sol intercanvi per torn. En cas que el jugador disposi del 7 i del 2 del mateix coll que el triomf, podria, primer, realitzar l'intercanvi amb la carta 7 i, tot seguit, l'intercanvi amb la 2.

Aquesta regla dona lloc a múltiples estratègies i tàctiques, sobretot en el joc del Tute on, per exemple, si un jugador disposa del rei de copes, i el triomf és el cavall del mateix coll, pot esperar al moment oportú per intercanviar el cavall i cantar “les 40”.

4.3.2. Les 10 d'últimes

Quan aquesta regla està activada, el jugador que guanya l'última ronda sumarà 10 punts addicionals a la seva puntuació.

Aquesta regla dona lloc a diverses estratègies i tàctiques. Per exemple, un jugador pot jugar les seves cartes quan s'apropa el final de la partida de certa manera que s'asseguri que s'emportarà l'última ronda, i, així, aconseguir els 10 punts addicionals, que poden ser determinants en el resultat de la partida.

4.3.3. Mà negra

La mà negra es dona quan un jugador disposa a la seva mà de l'As, el 3 i el rei del coll del triomf. El jugador que obtingui la mà negra guanyarà immediatament la partida amb la puntuació màxima, independentment de la puntuació de cada jugador en aquell moment. A més, la resta de jugadors no sumaran cap punt en aquella partida.

En aquest cas, un jugador que estigui perdent la partida per puntuació pot arribar a invertir la situació si guarda aquestes cartes i espera a tenir un cop de sort i aconseguir reunir les 3 cartes necessàries. Evidentment, no és una estratègia guanyadora, pel fet que és un cas poc freqüent, però tots els jugadors, inclús els que estan guanyant, l'han de tenir en compte.

Per exemple, en el joc del Tute on els jugadors estan obligats a assistir, si un jugador que està guanyant sospita que un dels rivals pot estar intentant reunir la mà negra, podria jugar una carta de triomf expressament per tal d'obligar-lo a tirar una d'aquestes cartes.

4.3.4. Caça del 3

Aquesta regla només s'aplica a partides de 2 jugadors i no té cap efecte amb més jugadors. Es dona quan el jugador inicial de la ronda juga el 3 del coll del triomf i el rival juga l'As del mateix coll, guanyant la ronda.

Quan això passa, es diu que el jugador “ha caçat el 3”, i el jugador que ha jugat l'As guanyarà automàticament la partida amb la puntuació màxima, independentment de la puntuació de cada jugador en aquell moment. A més, el jugador rival no sumarà cap punt en aquella partida.

En qualsevol de les rondes, sempre s'ha de donar el cas que primer es jugui el 3 i després l'As per considerar-se caçat, amb l'única excepció a l'última ronda, on el jugador pot jugar primer l'As i el rival estar obligat a tirar el 3 que li queda a la mà. Aquest cas representa que el jugador que té l'As ha arraconat al jugador que té el 3 i ha aconseguit “caçar el 3” al final de la partida.

Les estratègies i tàctiques amb aquesta regla són molt vàries, i, tot i no ser òptimes per a guanyar múltiples partides, poden ser determinants en alguns casos, sobretot en el joc del Tute on el rival està obligat a assistir i muntar la carta inicial de la ronda.

4.3.5. Només assistir

El Tute, amb totes les seves restriccions a l'hora de jugar una carta, és un joc on els jugadors estan molt condicionats amb la presa de decisions. L'estratègia es veu limitada, bàsicament, al jugador inicial d'una ronda, que és qui realment pot decidir quina carta vol utilitzar. La resta de jugadors estan limitats per les restriccions.

Aquesta regla elimina les restriccions de muntar, fallar i trepitjar, i, en conseqüència, només cal aplicar la restricció d'assistir. Amb aquest canvi s'aconsegueix que la resta de jugadors tinguin més capacitat d'estratègia, i puguin triar quina carta volen fer servir en cas que no disposin de cap carta del coll de ronda (poden reservar les cartes del coll del triomf, poden decidir no guanyar una ronda jugant una carta del mateix coll de ronda però amb menys valor...).

Aquesta regla és aplicada a la meva família i no l'he vista a cap aplicació, així que probablement és inventada, però s'ha afegit per aquest component estratègic diferent.

5. Conceptes tècnics

5.1. Entorn i estat del joc

En el camp de l'aprenentatge automàtic, l'entorn i l'estat del joc són conceptes fonamentals.

L'entorn fa referència a tot el conjunt de condicions i regles dins del qual opera un agent. En el cas d'aquesta investigació cadascun dels jocs als quals els agents hauran d'aprendre a jugar.

L'estat del joc és una representació concreta d'una situació dins d'aquest entorn per a un moment determinat i inclou tota la informació rellevant que un agent necessita per prendre una decisió. En un joc de cartes, l'estat del joc pot incloure les cartes que un jugador té a la seva mà o les cartes jugades fins al moment a la ronda actual.

Amb l'objectiu de guanyar la partida, un jugador prendrà una decisió o una altra segons la informació de la qual disposi a cada jugada.

5.2. Aprenentatge supervisat

Les tècniques d'aprenentatge supervisat utilitzen dades etiquetades per a l'entrenament d'un model que serà capaç de realitzar prediccions. Els algoritmes reben una sèrie d'exemples (conjunt de dades) que contenen tota la informació coneguda del problema que es vol resoldre junt amb la sortida que indica la resposta desitjada per a cadascun dels exemples.

L'objectiu és que el model pugui aprendre una funció que associï les dades d'entrada dels exemples del conjunt de dades amb les sortides esperades, de manera que pugui generalitzar aquesta associació a noves dades d'entrada que no hagi vist durant el seu entrenament.

Normalment, el conjunt de dades es divideix en tres subconjunts disjunts: entrenament, proves i validació. El d'entrenament serveix perquè la xarxa neuronal ajusti els seus valors i s'adapti als patrons i les relacions existents entre les dades d'entrada i la sortida associada a cadascun dels exemples. El de proves s'utilitza per avaluar els hiperparàmetres del model i el seu rendiment durant el procés d'entrenament. El de validació serveix per avaluar la capacitat de generalització del model a noves dades que no ha vist abans.

És important que els conjunts siguin disjunts, pel fet que l'ajustament dels hiperparàmetres i l'avaluació de la capacitat de generalització es durà a terme amb dades no utilitzades durant l'entrenament.

Existeixen múltiples tècniques d'aprenentatge supervisat, com la tècnica dels veïns més propers (KNN) o les màquines de vectors de suport (SVM). No obstant això, per aquesta investigació, s'utilitzarà les xarxes neuronals, on centenars o milions de neurones treballen juntes per a processar informació i executar tasques complexes. Aquesta tècnica és capaç d'associar relacions complexes entre les dades d'entrada i la seva capacitat de generalització les fan ideals per al problema que es vol resoldre.

La neurona és la unitat més bàsica de la xarxa neuronal i efectua operacions matemàtiques sobre les dades que rep. S'organitzen en diferents capes que reben un nom o un altre segons la seva posició dins de la xarxa neuronal. La capa d'entrada és la que rep les dades inicials, i la sortida de les seves neurones proporciona les dades d'entrada de la següent capa. Les capes que reben les sortides de les neurones d'una capa anterior i que proporcionen la seva sortida a una nova capa són anomenades capes intermèdies. La capa de sortida és l'encarregada de produir les sortides finals de la xarxa neuronal.

Cada neurona pot estar connectada amb una o diverses neurones de capes anteriors i posteriors. Cadascuna d'aquestes connexions té un pes associat que determina la importància

relativa d'aquesta connexió en el càlcul de la sortida final de la neurona, a més, cada neurona té un valor associat, anomenat biaix, que s'utilitza per acabar d'ajustar la sortida, a la qual se li aplica una funció d'activació que produeix el valor final de la sortida de la neurona.

En general, com més capes i connexions té una xarxa neuronal, més complexes poden ser les representacions que pot aprendre, però també s'incrementa el temps d'entrenament pel fet d'augmentar el total de càlculs que cal realitzar.

L'entrenament d'una xarxa neuronal segueix les següents fases:

1. Definició de l'arquitectura. Es defineix l'arquitectura de la xarxa neuronal (capes, neurones, funcions d'activació, sortida...) i s'inicialitzen de forma aleatòria els pesos i biaixos.

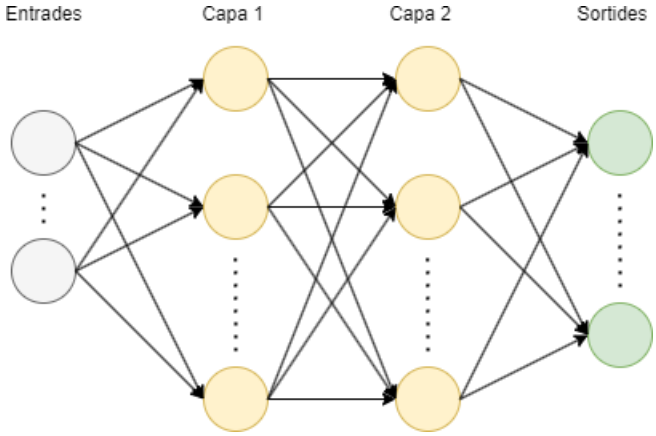

Figura 4: Xarxa neuronal totalment connectada amb dues capes

2. Propagació cap endavant (forward propagation). S'introdueixen les dades d'entrada a la xarxa neuronal i es propaga el càlcul de les neurones per la resta de capes de la xarxa neuronal, des de la capa d'entrada fins a la final. Cada neurona calcula la seva sortida sumant el pes ponderat de cada entrada amb el pes de la connexió, afegint el valor del seu biaix. A aquest valor calculat se li aplica una funció d'activació que acaba d'ajustar el valor.

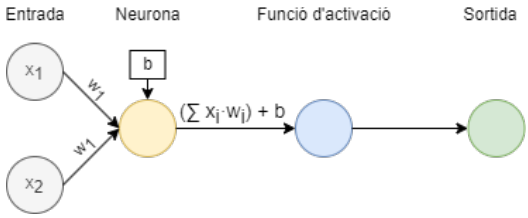

Figura 5: Progagació cap endavant d'una neurona

3. Càlcul de l'error de la sortida. Es compara la predicció amb la sortida esperada associada a l'exemple utilitzant una funció de pèrdua (loss), que calcula la distància entre la predicció i la sortida real.
4. Retropropagació (backpropagation). Es propaga l'error calculat en la fase anterior des de la sortida fins a la capa d'entrada i es calculen les contribucions de cadascun dels pesos a l'error real, amb l'objectiu de trobar la direcció en la qual s'han d'actualitzar els pesos i baixos per tal que la funció de pèrdua assoleixi un mínim.
5. Actualització dels pesos i biaixos. S'ajusten els pesos i biaixos de les neurones segons la contribució de cadascuna amb l'error de la predicció.
6. Iteració. Les fases es repeteixen de la fase 2 a la 5 per a la resta d'exemples disponibles en el conjunt de dades escollit per a l'entrenament.

La llibreria “Tensorflow” conté una sèrie de funcions que permeten aplicar l'entrenament descrit anteriorment, així com diverses funcions d'activació, càlcul de l'error, optimització i mètriques per dur a terme tota mena d'entrenaments amb aquest mètode.

5.3. Aprenentatge per reforç

Els algoritmes d'aprenentatge per reforç són tècniques d'aprenentatge automàtic on un agent aprèn a prendre decisions a mesura que interactua amb l'entorn. Es basen en la idea que l'agent millora mitjançant la retroalimentació que rep de l'entorn en forma de recompenses positives o negatives segons les seves accions.

Es poden diferenciar dues branques dins d'aquests algoritmes: les tècniques sense model (model-free) i les basades en model (model-based). Les primeres no necessiten construir un model de l'entorn, ja que l'agent defineix la seva política a partir de l'experiència acumulada, mentre que en les segones l'agent crea un model de l'entorn que inclou les probabilitats de transició entre estats i les recompenses associades a aquestes transicions. [16]

La Brisca i el Tute són jocs on no és possible predir el següent estat, ja que depèn de factors com la carta jugada pel rival o la carta robada al final del torn, per tant, és un problema que no disposa de cap model.

Q-Learning, SARSA, Deep Q-Networks i Monte Carlo són exemples de tècniques que no necessiten model. Però, aquesta investigació se centrarà només en la tècnica Monte Carlo, perquè utilitza simulacions i mostrejos aleatoris per estimar valors basats en la mitjana de la recompensa acumulada fins al final de l'episodi.

L'entrenament aplicant el mètode Monte Carlo segueix les següents fases [17]:

1. S'inicialitza una taula Q amb els seus valors a zero per a totes les parelles “estat-acció” i una taula política amb accions escollides aleatòriament.
2. S'inicialitza una taula Q amb totes les parelles “estat-acció” de l'entorn inicialitzada amb zeros i una taula política inicialitzada amb accions escollides aleatòriament.
3. Se simula un episodi on l'agent interactua amb l'entorn, i seleccionant accions de la política. Cada acció és recompensada i emmagatzemada junt amb l'estat i l'acció executada a la seva memòria.
4. Al final de l'episodi, es calcula el valor del retorn per a cada estat visitat utilitzant les recompenses rebudes, des que es visita l'estat fins al final de l'episodi, mitjançant la següent fórmula:

$$\left(G_t = R_{t+1} + \gamma \times R_{t+2} + \gamma^2 \times R_{t+3} + ... + \gamma^{T-t-1} \times R_t\right)$$

Figura 6: Fórmula del càlcul del retorn d'un estat

   “t” correspon al temps de l'estat pel qual es vol calcular el valor de retorn, “R_{t+x}” són totes les recompenses rebudes en accions posteriors fins al final de l'episodi i “γ” és el factor de descompte, que determina com és d'important la recompensa segons el moment en el qual succeeix l'acció (com més baix és el seu valor, més important el valor de retorn a les jugades inicials, mentre que com més proper a 1, s'iguala la importància de les recompenses al llarg de tot l'episodi, independentment de si l'acció succeeix a l'inici o al final).
5. S'actualitza la taula Q amb els retorns calculats en la fase anterior. Aquesta actualització es pot realitzar amb el mètode de la primera visita i el mètode de cada visita. En el primer, el valor esperat de cada parella “estat-acció” es calcula la primera vegada que l'agent visita l'estat i escull aquella acció, mentre que en la segona, es calcula la mitjana del valor esperat per aquell estat i acció durant tot l'entrenament.
6. S'actualitza la política de l'agent seleccionant l'acció que té un valor esperat més alt per a cada estat visitat durant l'episodi (mètode greedy) o amb una acció aleatòria si un valor a l'atzar és inferior al valor d'exploració “ε”, que decreix amb el pas dels episodis (mètode ε-greedy).
7. Es repeteixen les fases de l'1 a la 5 fins a finalitzar tots els episodis definits.

En el mètode de cada visita, tant el valor esperat com la política de l'agent tendeixen a convergir cap a valors òptims quan els episodis tendeixen a infinit, però aquest valor dependrà de la complexitat del problema que es vol resoldre.

5.4. Aprenentatge genètic

Les tècniques d'aprenentatge genètic estan inspirades en l'evolució biològica dels éssers vius i són utilitzades per a resoldre problemes d'optimització i cerca. Es basen en els conceptes de la selecció natural, reproducció i mutació de l'evolució per a solucionar de forma eficient problemes complexos.

Els cromosomes o individus són les estructures que representen les possibles solucions del problema, com cadenes de bits, vectors numèrics, estructures d'arbre, xarxes neuronals o altres representacions adequades al problema que es vol resoldre.

Les unitats bàsiques d'informació dels cromosomes són els gens i representen una característica o component específic de la solució. Per exemple, si el cromosoma està representat com a cadenes de bits, cada bit correspon a un gen diferent, i, per a xarxes neuronals, el gen podria ser representat pels pesos i biaix d'una neurona.

A un conjunt de cromosomes se l'anomena població i a cada iteració de l'algoritme generació (cada generació té la seva pròpia població, que representa l'evolució dels cromosomes al llarg del temps).

L'entrenament aplicant el mètode d'aprenentatge genètic segueix les següents fases:

1. Selecció de la població inicial. Es genera una població aleatòria.
2. Avaluació de l'aptitud dels cromosomes de la població. S'avalua cada cromosoma amb l'objectiu de quantificar la solució. Pot variar segons el problema que es vol resoldre.
3. Selecció de les millors solucions. Se seleccionen les solucions més aptes basant-se en l'avaluació de la fase anterior.
4. Encreuament (Crossover). Els individus més aptes intercanvien els seus gens per a generar una nova descendència amb propietats combinades dels seus pares.
5. Mutació (Mutation). La descendència és sotmesa a mutacions dels seus gens per incrementar la diversitat genètica de la nova població.
6. Reemplaçament. La descendència substitueix als individus menys aptes de la població, o, fins i tot, la seva totalitat.
7. Iteració. El procés es repeteix de la fase 2 a la fase 6 tantes generacions com calgui.

Aquest procés evolutiu a través de les generacions permet que les solucions més aptes es propaguin i s'apropin progressivament a solucions òptimes o properes per al problema que es vol resoldre.

6. Metodologia

6.1. Introducció

Després de la investigació sobre projectes existents d'implementació dels jocs per a la Brisca i el Tute, així com de projectes d'entrenament d'agents per aquests jocs, s'ha arribat a la conclusió que cap d'ells compleix els requisits inicials per al desenvolupament d'aquesta investigació. Per tant, es desenvoluparà l'entorn del joc, la implementació gràfica i el marc de treball per a l'entrenament des de zero.

S'utilitzarà el llenguatge de programació “Python” per desenvolupar tots els mòduls necessaris. Aquesta decisió s'ha pres considerant que l'entrenament es durà a terme amb la llibreria “TensorFlow” [20], que està suportada i integrada amb “Python”. Això facilitarà la integració de tots els mòduls i permetrà reutilitzar el codi.

6.2. Implementació de l'entorn

L'entorn del joc ha de permetre simular partides de la Brisca i el Tute, i, per tant, és necessari definir tots els components clau d'una partida. Aquests incloent les cartes i els colls disponibles, els jugadors i les seves mans, l'estat de cada ronda i l'estat de la partida (puntuació parcial, puntuació global, partides guanyades de cada jugador...).

Donada la semblança entre ambdós jocs, s'ha implementat un únic entorn que rep la informació necessària mitjançant paràmetres per a la gestió de les partides de cadascun dels jocs.

En els jocs de la Brisca i el Tute és molt important qui és el jugador que comença la primera ronda de les partides, ja que té un desavantatge inicial. Per aquest motiu, el jugador inicial es rota equitativament entre tots els jugadors.

Pel que fa a l'entrenament dels agents per a cada joc (Brisca, Tute i Tute amb l'obligació de només assistir), s'entrenarà un agent específic per a cada modalitat de joc (dos, tres i quatre jugadors i per equips). Això és necessari, ja que la informació de l'estat del joc varia per cada cas. Per exemple, hi ha més informació en la modalitat de quatre jugadors pel fet que hi ha més cartes en joc durant una ronda.

Per al desenvolupament de la GUI, s'utilitzarà Tkinter [21], una llibreria de Python que permet la creació d'interfícies gràfiques d'usuari d'una forma senzilla. Aquesta interfície permetrà analitzar el comportament dels agents durant les diferents situacions que poden sorgir al llarg de les partides.

6.3. Definició dels estats del joc

Les tècniques implementades necessiten conèixer la informació referent a l'estat del joc per poder prendre decisions per a cada situació.

Les variables que conformen l'estat del joc són les següents:

- Nombre de cartes restants per robar: indica com d'avançada està una partida. Més cartes implica que queden més rondes per jugar.
- Carta de triomf: indica quin coll té preferència durant el transcurs de la partida. És crucial per prendre decisions.
- Cartes jugades a la ronda: Les cartes que s'han jugat a la ronda influeixen en les decisions dels jugadors.
- Cartes a la mà del jugador: informació fonamental per la presa de decisions tant a curt com a llarg termini.
- Cartes conegudes dels rivals: en algunes ocasions, un jugador pot tenir informació sobre les que els seus rivals tenen a la mà.

- Cants dels jugadors (només Tute): Els cants poden influeixen en les decisions dels jugadors, especialment per obligar els rivals a jugar certes cartes com el rei o el cavall del coll del cant.
- Cartes jugades durant la partida: els jugadors experimentats porten un recompte de les cartes importants jugades al llarg la partida per poder jugar estratègicament.
- Puntuació parcial de la partida: els jugadors experimentats porten un recompte aproximat de la puntuació de tots els jugadors.
- Regles opcionals de la partida: Les diferents regles que s'apliquen a una partida influeixen en les decisions dels jugadors.

Els agents han de tenir accés a aquesta informació per ser capaços de prendre decisions en un moment donat de la partida. No obstant això, no sempre s'utilitzarà tota la informació per reduir la complexitat del problema.

A continuació, es detalla la representació utilitzada per a cadascuna de les variables.

6.3.1 Carta de triomf

Per a representar la carta de triomf es faran servir dues representacions utilitzant vectors binaris.

La primera farà servir una representació utilitzant vectors binaris de 14 posicions, on les 4 primeres corresponen a la codificació One Hot Encoding del coll de la carta (ors, bastos, espases i copes) i les 10 següents indiquen la preferència de la carta, també en codificació One Hot Encoding (la primera posició representa la carta de menor valor, 2, i l'última la de major valor, As).

A la taula 4 es poden veure exemples de cartes en aquesta representació.

	Vector One Hot Encoding del coll				**Vector One Hot Encoding del número de la carta**									
Carta	**Or**	**Bastos**	**Espases**	**Copes**	**2**	**4**	**5**	**6**	**7**	**10**	**11**	**12**	**3**	**As**
3 d'ors	1	0	0	0	0	0	0	0	0	0	0	0	1	0
3 de bastos	0	1	0	0	0	0	0	0	0	0	0	0	1	0
2 d'espases	0	0	1	0	1	0	0	0	0	0	0	0	0	0
12 de copes	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Taula 4: Exemples de codificació de la carta de triomf

La segona farà servir una representació utilitzant vectors binaris de 5 posicions amb la normalització de la preferència de la carta. Les 4 primeres posicions corresponen a la codificació One Hot Encoding del coll de la carta (ors, bastos, espases i copes) i la següent serà la preferència de la carta normalitzada per la diferència entre el màxim i el mínim (l'As és la carta més alta i val 10, i 2 és la carta més baixa i val 1).

6.3.2. Cartes jugades a la ronda

Per a representar les cartes jugades a la ronda es faran servir unes representacions similars que les utilitzades per a la carta de triomf.

Es farà servir una representació utilitzant vectors binaris de 14 o 5 posicions per a cada carta que està en joc. La diferència és que si alguna queden cartes per jugar, tot el seu vector serà “0”.

6.3.3. Cartes a la mà del jugador

Un jugador tindrà a la seva mà entre una i vuit cartes segons si el joc és la Brisca o el Tute.

Per a la Brisca, la representació de les cartes en el format de 14 bits fan necessaris 42 bits en total (3x14), en canvi, per al Tute són necessaris 112 bits (8x14). Però aquesta representació implica que hi ha un ordre en les cartes a la mà, i no es podrà diferenciar entre tenir el 3 d'ors, el 2 d'ors i el 4 d'ors que tenir el 4 d'ors, el 3 d'ors i el 2 d'ors (mateixes cartes però en diferent ordre). Per tant, aquesta representació no serveix.

Per a representar les cartes a la mà del jugador es farà servir una representació utilitzant vectors binaris de 40 posicions on cada posició representa una de les cartes disponibles.

Aquesta representació redueix el nombre de bits necessaris i resol el problema de l'ordre de les cartes a la mà.

6.3.4. Cartes conegudes dels rivals

Els jugadors poden conèixer alguna de les cartes dels seus rivals en moments molt específics de la partida, com quan es realitza un intercanvi de la carta de triomf o durant els cants al joc del Tute.

Per representar les cartes conegudes dels rivals es farà servir la mateixa representació que la utilitzada per a les cartes de la mà del jugador i es farà servir una representació utilitzant vectors binaris de 40 posicions per cada rival, on cada posició representa una de les cartes disponibles. Aquests estaran en ordre, des del jugador actual fins a la resta de jugadors en sentit horari.

Tot i que aquesta informació pot ser crucial en la presa de decisions, no és un cas que es doni amb gaire freqüència. Només en casos puntuals hi haurà un o dos bits actius per algun dels rivals, i la resta de situacions sempre seran “0”.

Per tant, la representació d'aquesta informació és molt ineficient i augmenta la complexitat de l'estat del joc, i s'ha optat per excloure-la de l'estat del joc en alguns dels experiments realitzats.

6.3.5. Cants dels jugadors (Tute)

Per representar els cants dels jugadors, s'utilitzarà una representació amb vectors binaris de 4 posicions per a cada jugador. Cada posició del vector correspon a la codificació One Hot Encoding del coll de la carta (ors, bastos, espases i copes). En el cas que un jugador no hagi realitzat cap cant, el vector binari sencer serà “0”.

Els vectors estaran en ordre, des del jugador actual fins a la resta de jugadors en sentit horari.

6.3.6. Cartes jugades al llarg de la partida i cartes restants per robar

Per representar les cartes jugades al llarg de la partida s'utilitzarà la mateixa representació que per a les cartes de la mà del jugador: un vector binari de 40 posicions, on cada posició representa una de les cartes disponibles.

Aquesta representació també permet conèixer quantes cartes queden a la baralla, ja que com més bits actius hi hagi, més avançada estarà la partida. D'aquesta manera es pot evitar representar explícitament les cartes restants per robar, pel fet que afegir aquesta informació seria redundant i no aporta cap valor addicional.

6.3.7. Puntuació parcial de la partida

La puntuació parcial de cada jugador pot arribar a un màxim de 130 punts a la Brisca amb la regla de “les 10 d'últimes” activada, i fins a 230 al Tute si un jugador aconsegueix fer tots els cants, també amb la regla de “les 10 d'últimes” activada.

La representació d'aquesta informació serà la puntuació de cada jugador discretitzada en 7 intervals disjunts: [0, 19], [20, 39], [40, 59], [60, 79], [80, 99], [100, 119], [120, 230]

Cada interval serà representat per un bit, on 1 indicarà l'interval de puntuació en el qual es troba cada jugador.

6.3.8. Regles opcionals de la partida

La representació de cada regla serà un bit, on 1 significa que la regla està activada i 0 desactivada.

Les regles aplicades seran “intercanvi de la carta de triomf”, “10 d'últimes”, “mà negre” i, en el cas de dos jugadors, també s'inclourà “caça del 3”.

S'ha considerat que la regla “només assistir” del Tute modifica tant la forma de jugar que necessita ser una modalitat de joc per si mateixa. Així, hi haurà un agent que aprengui a jugar al Tute sense la regla “només assistir”, aplicant una combinació de les altres regles opcionals, i un altre agent que apliqui la regla “només assistir” junt amb les altres regles opcionals.

6.4. Aprenentatge supervisat

La implementació realitzada per a l'aprenentatge supervisat es pot dividir en quatre parts: les diferents variants utilitzades per als models, la generació dels conjunts de dades, la selecció de les característiques del conjunt (informació d'entrada i etiqueta de sortida) i la divisió del conjunt inicial i selecció de l'arquitectura adequada, i els experiments realitzats fins a obtenir els models finals.

6.4.1. Variants dels models

S'han implementat tres variants dels models. El primer, utilitzant l'etiqueta de partida guanyada/perduda, és la variant “SL-WL” de “Win or Lose”. El segon, amb l'etiqueta de puntuació de ronda, és la variant “SL-P” de “Points”. I el tercer, utilitzant l'etiqueta resultant de la suma de la puntuació de la ronda i el valor d'una funció heurística, és la variant “SL-PH” de “Points + Heuristic”.

Variant “SL-WL”

La xarxa neuronal té una única sortida a la qual se li aplica la funció d'activació “sigmoid”, que converteix la sortida en un valor entre 0 i 1. Com més proper a 1 és la sortida, més probabilitat hi ha que l'acció escollida porti a la victòria; en canvi, com més propera a 0, més probabilitat que l'acció porti a la derrota.

Per escollir la millor acció que un model pot realitzar en un moment donat de la partida, es fa una predicció per cadascuna de les accions possibles durant el seu torn, i es triarà l'acció que la seva predicció s'apropi més a 1.

Variant “SL-P”

En la variant “SL-P”, la xarxa neuronal té tantes sortides com puntuacions diferents es poden donar. La sortida serà un vector binari en codificació One Hot Encoding ordenat per la puntuació de menor a major. Quan la puntuació és, per exemple, 20, el bit actiu correspondrà a la posició que li pertoqui, mentre que les restants seran 0.

A la sortida de la xarxa neuronal se li aplica la funció d'activació “softmax”, que converteix totes les sortides de la xarxa neuronal en un vector de probabilitats (la suma de tots els valors de la sortida és 1). Aquesta sortida indica la probabilitat que, per a una acció concreta, s'obtingui cadascuna de les possibles puntuacions.

Es triarà l'acció que tingui major probabilitat i que estigui situada més a la dreta del vector, ja que serà la que té més probabilitat d'aconseguir més punts. Per exemple, si per a l'acció 1 s'obté el següent vector (0.1, 0,2, 0.4, 0.2, 0.1) i per a la segona acció (0,1, 0.2, 0.2, 0.4, 0.1), es triarà la segona acció perquè és la que té més probabilitat i està més situada a la dreta.

Variant “SL-PH”

La variant “SL-PH” està implementada de la mateixa manera que la variant “SL-P”, amb la diferència que la sortida tindrà tants valors com combinacions diferents es poden donar de la suma de la puntuació i l'heurístic.

6.4.2. Generació del conjunt de dades

Per poder entrenar les xarxes neuronals, és necessari un conjunt de dades, però, com que no es disposa d'un conjunt inicial, es generarà un on les accions dels jugadors s'escolliran de manera aleatòria.

Cada fila contindrà la informació d'un torn d'un jugador, que inclou l'estat complet del joc i múltiples etiquetes de sortida que s'utilitzaran en diferents experiments per obtenir comportaments estratègics diferents:

- Els punts pel jugador en aquella ronda.
- El resultat final de la partida (guanya o perd la partida).
- El valor d'una funció heurística que avalua la jugada.

Com que el conjunt conté jugades aleatòries, moltes jugades que serien considerades dolentes poden aconseguir un alt nombre de punts, però davant d'un jugador professional serien jugades perdudes. La funció heurística té com a objectiu corregir aquest comportament.

El valor heurístic es calcula mitjançant una funció que avalua la qualitat de la jugada, puntuant positivament les que són considerades correctes i negativament aquelles que només funcionen contra rivals inexperts, tot i permetre guanyar molts punts.

S'emmagatzemen dues versions del conjunt de dades: una utilitzant la representació de l'estat del joc en format d'inputs binaris, apta per a l'entrenament de les xarxes neuronals directament; i una altra codificada, que redueix la mida dels fitxers emmagatzemats, encara que necessiti ser descodificat abans de cada entrenament, un procés que implica un temps addicional per a cada experiment.

En total, s'han generat 48 conjunts de dades diferents: quatre per a cadascuna de les modalitats dels jocs amb diferent quantitat de partides (1.000, 1.500, 3.000 i 8.000 partides), amb cada combinació de regles opcionals. Amb dos jugadors hi ha 4 regles, creant 2^4 combinacions diferents; per a tres i quatre jugadors (incloent-hi equips) hi ha tres regles generant 2^3 combinacions. Això resulta en un total de partides (ps) que varia segons la combinació de modalitat de joc i nombre de partides, com es detalla a la taula 5, juntament amb el temps requerit (t) per a la generació de cada conjunt.

Inicialment, s'ha intentat generar un conjunt amb 10.000 partides, però la limitació de memòria RAM de l'equip ha obligat a reduir-ho fins a 8.000 partides per a completar l'entrenament dels models sense problemes.

Partides / modalitat		1000		1500		3000		8000	
		ps	t	ps	t	ps	t	ps	t
Brisca	**2j**	16000	84s	24000	128s	48000	4m	128000	19m
	3j	8000	44s	12000	67s	24000	2m	64000	10m
	4j	8000	51s	12000	78s	24000	3m	64000	11m
	equips	8000	52s	12000	79s	24000	3m	64000	12m
Tute	**2j**	16000	2m	24000	205s	48000	7m	128000	28m
	3j	8000	65s	12000	99s	24000	3m	64000	14m
	4j	8000	72s	12000	103s	24000	4m	64000	15m
	equips	8000	70s	12000	102s	24000	4m	64000	15m
Tute assistir	**2j**	16000	2m	24000	203s	48000	7m	128000	30m
	3j	8000	65s	12000	98s	24000	3m	64000	14m
	4j	8000	72s	12000	101s	24000	4m	64000	16m
	equips	8000	69s	12000	102s	24000	4m	64000	16m

Taula 5: Total de partides generades i temps d'execució per a la generació dels conjunts inicials

6.4.3. Selecció de característiques, divisió del conjunt de dades i arquitectura

No tota la informació emmagatzemada en els conjunts de dades ha estat utilitzada per entrenar els models. Cada experiment realitzat selecciona un dels conjunts i utilitza tot o part de la informació que contenen, tant de les dades d'entrada com de les etiquetes de sortida, amb l'objectiu d'obtenir models que implementin diferents estratègies.

Per exemple, un model entrenat amb l'etiqueta de puntuació per ronda buscarà maximitzar els punts a cada jugada, mentre que un model que utilitzi l'etiqueta del valor heurístic buscarà optimitzar les jugades segons el criteri d'aquest valor.

Per a cada experiment, es realitza una divisió del conjunt de dades en tres subconjunts disjunts: entrenament (80%), validació (10%) i proves (10%).

A més de la divisió de dades, per obtenir models precisos, eficients i equilibrats s'ha de seleccionar de forma adequada el nombre de capes i neurones de la xarxa neuronal, la funció d'activació per a cadascuna de les capes i la funció d'optimització.

Per exemple, una xarxa neuronal amb poques capes i neurones patirà de subajustament, ja que no serà capaç de representar la complexitat de les dades. D'altra banda, si té massa capes i neurones serà més precisa, però molt més lenta en el seu entrenament.

6.4.4. Experiments i entrenament dels models

S'han dut a terme múltiples experiments per ajustar paràmetres com el nombre de partides del conjunt inicial, la informació de les dades d'entrada, l'etiqueta de sortida i l'arquitectura de la xarxa neuronal.

A continuació es detalla cadascun dels experiments.

Primer experiment

En aquest primer experiment s'han explorat les variants “SL-WL” i “SL-P” per a la modalitat de dos jugadors de la Brisca. S'ha utilitzat el conjunt de 1.000 partides i totes les entrades han estat representades en format binari sense normalitzar.

La taula 6 mostra el total d'entrades d'aquesta configuració per totes les modalitats.

		Triomf	Cartes ronda	Cartes mans	Cants	Cartes jugades	Puntuació	Regles	Accions	Total
Brisca	**2j**	14	14	80	0	40	7	4	41	200
	3j	14	28	120	0	40	7	3	41	253
	4j	14	42	160	0	40	7	3	41	307
	equips	14	42	160	0	40	7	3	41	307
Tute / Tute només assistir	**2j**	14	14	80	8	40	7	4	45	212
	3j	14	28	120	12	40	7	3	45	269
	4j	14	42	160	16	40	7	3	45	327
	equips	14	42	160	16	40	7	3	45	327

Taula 6: Total d'entrades en el primer experiment de l'entrenament supervisat

La xarxa neuronal consta de tres capes amb un total de neurones que depenen del nombre de bits de l'entrada: “inputs/2”, “inputs/4” i “inputs/6” respectivament. S'ha utilitzat l'optimitzador “adam” i la funció d'activació “relu” a totes les capes. Per exemple, per a la modalitat de la Brisca per a 2 jugadors, les capes tindran 100, 50 i 33 neurones respectivament, donat que es fan servir 200 bits per aquesta modalitat específica.

Per a la variant “SL-P” s'ha utilitzat la puntuació per ronda positiva quan el jugador guanya la ronda o negativa quan la perd.

Els models obtinguts han necessitat 10 minuts d'entrenament cadascun, i han estat observats visualment en partides, per comprovar la seva capacitat en la presa de decisions.

El model de la variant “SL-WL” ha mostrat inconsistències significatives, prenent decisions tan bones com dolentes en situacions similars. Això es deu a la naturalesa aleatòria de les accions en el conjunt de dades, que inclou partides amb jugades molt variables.

D'altra banda, el model de la variant “SL-P” ha demostrat una millor coherència en la presa de decisions, ja que busca maximitzar la puntuació a cada ronda. Tot i això, també ha mostrat inconsistències en altres situacions.

S'ha repetit l'experiment amb el conjunt de 3.000 partides per descartar que el problema sigui que el conjunt inicial disposa de poques dades. Tot i incrementar el temps d'entrenament a 20 minuts, els resultats no han millorat.

Finalment, s'ha entrenat la resta de models per a totes les modalitats utilitzant la mateixa configuració amb el conjunt de 1.000 partides, però només amb la variant “SL-P”.

Els models entrenats per a les modalitats dos jugadors del Tute mostren una estratègia similar al de la modalitat de la Brisca, i també intenten maximitzar la puntuació a cada ronda, però, els models per a les modalitats de més de dos jugadors no segueixen cap estratègia i presenten un comportament molt erràtic i inconsistent.

Els models entrenats per a les modalitats dos jugadors del Tute mostren una estratègia similar al de la modalitat de la Brisca i també intenten maximitzar la seva puntuació a cada ronda, però, els models per les modalitats de més de dos jugadors no segueixen cap estratègia i presenten un comportament molt erràtic i inconsistent.

Es creu que aquest error prové de tractar la puntuació de la ronda com a negativa quan el jugador no guanya una ronda. Aquesta decisió no afecta la modalitat de dos jugadors, on els punts negatius i positius estan en equilibri. En canvi, per a més de dos jugadors, aquest equilibri es perd.

Segon experiment

En aquest segon experiment, s'ha utilitzat la mateixa configuració anterior per la modalitat de dos jugadors de la Brisca, però amb un canvi important: s'ha eliminat la puntuació negativa de l'etiqueta de sortida del conjunt de dades. Ara, l'etiqueta pren el valor de “0” quan era negativa, simplificant també el total de sortides de la xarxa neuronal.

Els models continuen maximitzant la puntuació a cada, però ara mostren un comportament més agressiu quan són els qui comencen una ronda. L'eliminació de la puntuació negativa fa que el model consideri una jugada on perdia “20” punts com una jugada normal, ja que ara no en perd ni en suma cap, sinó que obté “0” punts.

S'ha repetit l'experiment per a la resta de jocs i modalitats, però no s'ha observat cap millora en les decisions dels models per a les modalitats de més de dos jugadors. No sembla que aquesta variant sigui adequada per aquestes modalitats.

Tercer experiment

En aquest tercer experiment, s'ha utilitzat la mateixa configuració anterior per a la modalitat de dos jugadors de la Brisca i el Tute, i s'ha introduït una funció heurística per guiar les decisions dels models.

S'han observat partides dels models obtinguts i s'ha pogut constatar que la seva estratègia és coherent en la majoria de les situacions, intentant maximitzar la puntuació de cada ronda, però tenint en compte la funció heurística. Malgrat això, a vegades les seves decisions són massa previsibles.

Per acabar l'experiment, s'han entrenat els models per a la resta de modalitats i s'ha pogut confirmar que tots segueixen la mateixa estratègia de maximitzar punts tenint en compte la funció heurística, però, com que la funció està optimitzada per a la modalitat de dos jugadors, es poden observar algunes incoherències en algunes situacions concretes que no s'ha tingut en compte.

Quart experiment

En aquest experiment, s'han entrenat diversos models utilitzant la mateixa configuració que en l'experiment anterior, però introduint una reducció en el nombre de capes i neurones de la xarxa neuronal. L'objectiu és determinar fins a quin punt és possible disminuir l'arquitectura de la xarxa sense perdre rendiment.

Els models han estat entrenats per a la modalitat de dos i quatre jugadors de la Brisca. S'ha utilitzat una arquitectura d'una capa amb dues neurones de control, per confirmar que les decisions preses no són fruit de l'atzar, una capa amb 50 neurones, dues capes amb 50 i 25 neurones i dues capes de 75 i 50 neurones.

S'han simulat 150 partides, on cada model ha estat emparellat amb el model de tres capes per a la modalitat de la Brisca per dos jugadors. La taula 7 mostra els resultats de la simulació on el primer resultat indica el nombre de victòries del model de 3 capes i el segon el de cada model.

Neurones	**2**	**50**	**50-25**	**75-50**
Resultat	112 - 39	82 - 71	84 - 67	76 - 74

Taula 7: Resultats de la simulació de 150 partides entre models amb reducció de l'arquitectura. Brisca - dos jugadors

El model de dues capes de 75 i 50 neurones ha aconseguit igualar les victòries del model de tres capes. No obstant això, s'ha observat que a mesura que es redueix el nombre de neurones

i capes, el rendiment tendeix a disminuir lleugerament. És sorprenent que el model amb una sola capa de 50 neurones hagi superat al model de dues capes de 50 i 25 neurones.

Per a la modalitat de quatre jugadors s'han simulat 300 partides, variant les posicions dels jugadors per garantir la imparcialitat en els resultats (no és el mateix ser el jugador exactament anterior, posterior o entremig d'un altre jugador). La taula 8 mostra el resultat de la simulació, on el primer resulta és el del model de tres capes.

Neurones	**50 / 50-25 / 75-50**	**75-50 / 50-25 / 50**	**50-25 / 75-50 / 50**	**Totals 75-50 / 50-25 / 50**
Resultat	82 - 84 - 69 - 68	89 - 60 - 86 - 74	85 - 78 - 74 - 67	256 - 202 - 233 - 225

Taula 8: Resultats de la simulació de 300 partides entre models. Brisca - quatre jugadors - quart experiment

Tot i que els models estan igualats, sembla que la configuració de tres capes funciona lleugerament millor que la de dues capes amb 50 i 25 neurones.

Cinquè experiment

El cinquè experiment se centra en la reducció de la dimensionalitat de les dades d'entrada de la xarxa neuronal. En els experiments anteriors s'ha constatat que la quantitat d'entrades per a les cartes de les mans dels rivals és molt elevada (40 inputs binaris per cada rival), tot i que en la pràctica la majoria de les cartes són desconegudes (tots els inputs són 0 en el 99% de les jugades). Per tant, s'ha decidit eliminar aquesta informació. La taula 9 mostra els inputs que s'utilitzaran després de l'eliminació per a cada modalitat.

		Triomf	**Cartes ronda**	**Cartes mans**	**Cants**	**Cartes jugades**	**Puntuació**	**Regles**	**Accions**	**Total**
Brisca	**2j**	14	14	80 → 40	0	40	7	4	41	200 → 160
	3j	14	28	120 → 40	0	40	7	3	41	253 → 173
	4j	14	42	160 → 40	0	40	7	3	41	307 → 187
	equips	14	42	160 → 40	0	40	7	3	41	307 → 187
Tute / Tute només assistir	**2j**	14	14	80 → 40	8	40	7	4	45	212 → 172
	3j	14	28	120 → 40	12	40	7	3	45	269 → 189
	4j	14	42	160 → 40	16	40	7	3	45	327 → 207
	equips	14	42	160 → 40	16	40	7	3	45	327 → 207

Taula 9: Total d'entrades amb eliminació de les cartes de les mans dels rivals

A més, els experiments anteriors també han suggerit que es pot simplificar la representació de la carta de triomf i de les cartes jugades a la ronda. En lloc d'utilitzar un vector de 14 posicions es pot utilitzar la versió normalitzada, que utilitza un vector de 5 posicions. La taula 10 detalla com quedaran les entrades amb aquesta normalització.

		Triomf	**Cartes ronda**	**Cartes mans**	**Cants**	**Cartes jugades**	**Puntuació**	**Regles**	**Accions**	**Total**
Brisca	**2j**	14 → 5	14 → 5	40	0	40	7	4	41	160 → 142
	3j	14 → 5	28 → 10	40	0	40	7	3	41	173 → 146
	4j	14 → 5	42 → 15	40	0	40	7	3	41	187 → 151
	equips	14 → 5	42 → 15	40	0	40	7	3	41	187 → 151
Tute / Tute només assistir	**2j**	14 → 5	14 → 5	40	8	40	7	4	45	172 → 154
	3j	14 → 5	28 → 10	40	12	40	7	3	45	189 → 162
	4j	14 → 5	42 → 15	40	16	40	7	3	45	207 → 171
	equips	14 → 5	42 → 15	40	16	40	7	3	45	207 → 171

Taula 10: Total d'entrades amb la normalització de la carta de triomf i les cartes de la ronda

S'ha realitzat un entrenament per a les modalitats de dos i quatre jugadors de la Brisca amb aquestes noves entrades, utilitzant una configuració de dues capes de 75 i 50 neurones, que s'ha mostrat efectiva en l'experiment anterior.

Després d'observar diverses simulacions de partides entre els models obtinguts, s'ha confirmat que continuen sent eficients i segueixen l'estratègia de la seva etiqueta.

Per concloure aquest experiment, s'ha considerat també normalitzar el vector de les cartes de la mà del jugador, les cartes jugades durant tota la partida i les regles aplicades. L'objectiu és obtenir un valor enter corresponent a la representació binària d'aquesta informació i normalitzar-ho per la diferència entre el màxim i el mínim. La taula 11 mostra com seran les entrades amb aquesta última normalització.

		Triomf	Cartes ronda	Cartes mans	Cants	Cartes jugades	Puntuació	Regles	Accions	Total
Brisca	**2j**	5	5	40 → 1	0	40 → 1	7	4 → 1	41	142 → 61
	3j	5	10	40 → 1	0	40 → 1	7	3 → 1	41	146 → 65
	4j	5	15	40 → 1	0	40 → 1	7	3 → 1	41	151 → 70
	equips	5	15	40 → 1	0	40 → 1	7	3 → 1	41	151 → 70
Tute / Tute només assistir	**2j**	5	5	40 → 1	8	40 → 1	7	4 → 1	45	154 → 73
	3j	5	10	40 → 1	12	40 → 1	7	3 → 1	45	162 → 82
	4j	5	15	40 → 1	16	40 → 1	7	3 → 1	45	171 → 91
	equips	5	15	40 → 1	16	40 → 1	7	3 → 1	45	171 → 91

Taula 11: Total d'entrades amb la normalització de les cartes de la mà i les cartes jugades en tota la partida

Amb aquesta reducció de la dimensionalitat, s'ha dut a terme l'entrenament per a les modalitats de dos i quatre jugadors de la Brisca amb dues capes de 50 i 25 neurones. En les simulacions de partides dels models s'ha observat els models mantenen el rendiment dels experiments anteriors, validant l'eficàcia de la reducció de la dimensionalitat implementada.

Sisè experiment

El sisè experiment se centra en l'entrenament de models utilitzant un conjunt de dades generat pels mateixos models obtinguts en els experiments anteriors, en comptes de triar accions aleatòries. L'objectiu és disposar d'un conjunt de dades que reflecteixi jugades de jugadors més "experimentats", amb menys jugades dolentes comparades amb els conjunts utilitzats prèviament.

La generació d'aquest nou conjunt de dades ha requerit un temps considerable, ja que l'ús dels models per a les prediccions té un cost computacional molt superior al de la tria d'accions aleatòries. S'han necessitat més de quatre hores per generar un conjunt de 1.000 partides per la modalitat de dos jugadors de la Brisca.

Abans de procedir a generar els conjunts per a la resta de modalitats, s'ha entrenat el model per a la modalitat de dos jugadors de la Brisca utilitzant aquest nou conjunt per a les tres variants "SL-WL", "SL-P" i "SL-PH".

Després d'analitzar les simulacions i diverses partides amb cadascun dels models, s'ha observat que cap d'ells segueix una estratègia clara i eficaç. Contràriament a les expectatives, tots els models mostren un comportament erroni i inconsistent en la majoria de les situacions.

El principal problema és que el nou conjunt de dades no disposa de prou variabilitat en els exemples que conté. El conjunt reflecteix només accions que segueixen la lògica de la funció heurística, i això ha provocat que pateixin sobreentrenament (overfitting), és a dir, estan entrenats amb unes accions específiques i no poden generalitzar situacions que no eren presents en el conjunt d'entrenament.

Entrenament final

Per a l'entrenament final dels models, s'ha utilitzat la mateixa configuració que en el cinquè experiment, però amb els conjunts de dades que contenen 8.000 partides.

En total s'han entrenat dos models per a cadascuna de les modalitats (24 models), un utilitzant dues capes de 50 i 25 neurones, i un altre que utilitza tres capes amb un total de neurones que depenen del nombre de bits de l'entrada: “inputs/2”, “inputs/4” i “inputs/6” respectivament, ambdues utilitzant la variant “SL-PH”.

La taula 12 mostra el total de neurones per als models de tres capes per a cada modalitat.

		Total	Capa 1 (total / 2)	Capa 2 (total / 4)	Capa 3 (total / 6)
Brisca	**2j**	61	31	15	10
	3j	65	32	16	11
	4j	70	35	18	12
	equips	70	35	18	12
Tute / Tute només assistir	**2j**	73	37	18	12
	3j	82	41	21	14
	4j	91	46	23	15
	equips	91	46	23	15

Taula 12: Neurones dels models finals de tres capes

El temps d'entrenament dedicat a cada modalitat varia. Per a la modalitat de dos jugadors es requereix aproximadament 1 hora i 40 minuts per model, mentre que per a les modalitats de tres i quatre jugadors, així com per a la modalitat per equips, el temps és d'aproximadament 1 hora i 15 minuts per model.

6.5. Aprenentatge per reforç

La implementació realitzada per a l'aprenentatge per reforç es pot dividir en tres parts: el mètode Monte Carlo que s'ha utilitzat, les diferents variants d'agents utilitzades i els experiments realitzats fins a obtenir els agents finals.

6.5.1. Variants dels agents

Variant “MC-OS”

Aquesta variant utilitza un sol valor per a la clau dels seus diccionaris, que correspon al valor decimal dels inputs binaris de les entrades. És a dir, es codifica el valor binari resultant de totes les entrades en un únic valor decimal.

Les entrades utilitzades en aquesta variant són les mateixes que s'han fet servir en l'aprenentatge supervisat abans de la reducció de la dimensionalitat. Això inclou la carta de triomf, les cartes jugades a la ronda, les cartes a la mà de l'agent, els cants dels jugadors (només per al Tute), les cartes jugades durant tota la partida i les regles que s'estan aplicant.

Variant “MC-MS”

D'altra banda, la variant del mètode Monte Carlo “MC-MS” (Multiple State) utilitza una tupla de valors decimals per a la clau dels seus diccionaris. És a dir, es codifica el valor binari de cada informació de les entrades en un valor decimal separat.

Així, es tindrà un valor decimal per a cada tipus d'informació de l'estat del joc: la carta de triomf, les cartes jugades a la ronda, les cartes a la mà de l'agent, els cants dels jugadors (només per al Tute), les cartes jugades durant tota la partida i les regles que s'estan aplicant.

Aquesta tupla permet que l'agent pugui escollir una acció de la seva política en situacions que no ha vist durant el seu entrenament. Això es deu al fet que s'ha creat una abstracció de l'estat (minimitzant l'estat) per cercar situacions similars on sí que disposa d'informació.

6.5.2. Definició dels diccionaris

Per a l'entrenament dels agents amb aquest mètode seran necessaris els següents diccionaris. Tot i parlar d'estat, es fa referència tant a l'estat únic de la variant “MC-OS” com a la tupla de la variant “MC-MS”:

- Q: la clau d'aquest diccionari és la tupla “estat-acció”, i conté el valor esperat per a cada parella.
- Parelles visitades: la clau d'aquest diccionari també és la tupla “estat-acció”, i conté el total de vegades que s'ha visitat un estat i s'ha triat una acció “X”.
- Política: la clau d'aquest diccionari és l'estat, i conté la millor acció que l'agent pot dur a terme per a cada estat.
- Accions: la clau d'aquest diccionari també és l'estat, i conté les diferents accions que l'agent pot dur a terme per a una situació concreta.

La implementació realitzada difereix una mica del mètode original. En els jocs de la Brisca i el Tute, el nombre total d'estats és massa elevat i el seu valor numèric no és continu (per exemple, no existeixen els estats 1 ni 2).

Per tant, en comptes d'inicialitzar els diccionaris amb totes les claus existents amb un valor de 0 s'inicialitzaran buits, i es considerarà que el seu valor és 0 si la clau no existeix dins del diccionari (tant per als diccionaris Q com per les parelles visitades) i se seleccionarà una acció aleatòria per al diccionari de la política.

Aquesta decisió permet que les fases inicials de l'entrenament s'executin més ràpidament, pel fet que els diccionaris seran més lleugers que en fases posteriors de l'entrenament. A més, s'evita del càlcul de tots els possibles estats al començament de cada entrenament.

6.5.3. Actualització de la política

S'ha implementat el mètode de Monte Carlo amb actualització del valor esperat a cada visita.

Al començament de cada episodi, s'inicialitza una llista per a cada agent que conté la memòria amb la tupla “estat visitat-acció escollida-recompensa rebuda-possibles accions” per a cada torn de l'agent.

L'estat, com s'ha comentat anteriorment, és un valor enter o una tupla de valors enters segons la variant utilitzada. L'acció escollida correspon a l'acció que disposa en aquell moment la política per a l'estat actual, o una de les accions que l'agent pot dur a terme de forma aleatòria en cas que no existeixi aquell estat. La recompensa rebuda serà la puntuació de la ronda: positiva si l'agent ha guanyat la ronda i negativa si l'ha perduda.

La idea sobre la recompensa és que l'agent intenti maximitzar la puntuació al llarg de la partida (puntuació positiva), evitant jugar cartes que li facin perdre molts punts si pot evitar-ho (puntuació negativa). Per exemple, si el jugador rival s'ha emportat “20” punts i la recompensa de l'agent fos 0 (els punts que s'emporta a la ronda), aquest no contemplarà aquesta acció com negativa. En canvi, si la recompensa és de “-20” punts, l'agent intentarà que aquesta situació no es torni a repetir en el futur.

En finalitzar un episodi, es disposa d'una llista amb la memòria per a cada agent, que s'utilitzarà per a l'actualització de la política de cadascun dels agents. L'actualització de la política d'un agent comença amb el càlcul del retorn acumulat, G, per cada tupla estat-acció-recompensa que es disposa a la seva memòria, començant des de l'última jugada fins a la primera.

Aquesta estratègia permet que el càlcul del retorn per a cada estat sigui més senzill, ja que cada retorn d'un estat necessita conèixer el retorn acumulat dels estats futurs fins a la fi de l'episodi. Així, si es recorre la memòria de forma inversa, només cal calcular el retorn de cada estat utilitzant en el retorn acumulat de l'anterior.

A continuació, es procedeix a actualitzar el diccionari de parelles visitades. El valor per a la clau "estat-acció" s'inicialitza a 1 quan encara no ha estat definit, i se li suma 1 quan ja ho està.

Després, es procedeix a actualitzar el valor esperat per a cada parella "estat-acció" al diccionari Q. Com que s'ha implementat el mètode de Monte Carlo per a cada visita, s'ha de calcular la mitjana del valor esperat per cada "estat-acció" que s'ha visitat durant l'episodi, i això es pot fer utilitzant la fórmula de la mitjana incremental, que es pot veure a la fórmula 6.

$$Q_{new}[sa] = Q_{old}[sa] + \left(\frac{1}{\text{pairs\_visited}[sa]}\right) \times (G_{sa} - Q_{old}[sa])$$

Fórmula 7: Fórmula de la mitjana incremental

$Q_{new}[sa]$ és el nou valor esperat per la parella "estat-acció", $Q_{old}[sa]$ és el valor esperat previ a l'actualització, pairs_visited[sa] és les vegades que s'ha visitat un estat i s'ha fet una acció i G_{sa} és el retorn calculat en aquest episodi.

Per actualitzar la política s'ha utilitzat el mètode ε-greedy, que consisteix a inicialitzar un valor ε entre 0 i 1 a l'inici de l'entrenament, que s'utilitza per decidir si s'emmagatzema l'acció amb el valor esperat més alt a la política (el valor aleatori generat és més gran que ε) o una acció a l'atzar (el valor aleatori és inferior o igual a ε).

El valor d'exploració es disminueix cada vegada que finalitza un episodi, de manera que el mètode ε-greedy s'acaba convertint en un mètode greedy, on sempre s'escull la millor acció possible.

La implementació realitzada diferencia dos casos: un agent juga contra altres agents o contra si mateix. En el primer cas, diversos agents milloren a mesura que se simulen les partides, però l'exploració dels agents està limitada a la part de la partida que han vist (les seves jugades). En canvi, en el segon cas, l'agent sempre juga contra una versió actualitzada de si mateix de l'últim episodi, fet que li permet explorar més estats més ràpidament, ja que té una visió completa de tota la partida des del punt de vista de tots els jugadors.

L'actualització de la política per ambdós casos només difereix en el fet que, en les simulacions de partides entre diferents agents, cadascun actualitza els seus diccionaris a partir de la seva pròpia memòria. Mentre que l'agent que juga contra si mateix actualitza la política d'acord amb cadascuna de les memòries de tots els jugadors.

6.5.4. Experiments i entrenament dels agents

S'han dut a terme diversos experiments per provar diferents configuracions de paràmetres com el valor d'exploració (ε) i el seu decreixement, el valor de descompte (γ), i altres tècniques implementades.

Els agents només poden ser entrenats amb una configuració de regles opcionals a la vegada. Per tant, perquè aprenguin a jugar amb altres regles, caldrà reentrenar-los amb diferents combinacions o entrenar agents específics per a cada combinació.

Tots els experiments detallats s'han executat sense cap regla opcional i s'han mesurat en dies d'entrenament, però no són ininterromputs (24 hores seguides), sinó que cada dia equival a entre 8 i 12 hores.

Primer experiment

Aquest primer experiment serveix com a punt de contacte amb l'algoritme amb l'objectiu de descobrir el temps necessari per a l'entrenament en la modalitat de dos jugadors per a la Brisca.

S'ha utilitzat la variant “MC-OS” amb un valor d'exploració ε del 0,05, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001 amb un total de 500.000 episodis. El valor de decreixement està pensat perquè el valor d'ε arribi a 0 un cop finalitzin tots els episodis, per tant, no arribarà a convertir-se en un mètode greedy, i el valor de γ perquè cada jugada de la partida és igual d'important a l'hora de sumar la màxima puntuació possible.

En aquest experiment s'enfronten dos agents diferents utilitzant només la puntuació positiva de la ronda com a recompensa.

Durant l'entrenament dels agents, els diccionaris només es troben a la memòria RAM. Com que és el primer experiment, es desconeix si hi pot haver algun problema durant l'execució que provoqui la pèrdua de la seva informació. Per aquest motiu, els diccionaris s'emmagatzemen físicament després de cada 1.000 episodis.

Els primers 10.000 episodis s'han executat molt ràpidament (en qüestió de tres de minuts), però a mesura que s'han anat executant més episodis, aquest temps ha augmentat. En total s'han necessitat 11 hores per finalitzar l'entrenament.

Com que l'experiment s'ha deixat desatès, no s'ha pogut observar la causa d'aquest augment de temps fins a gairebé al final. Tot i que els diccionaris augmenten de mida, les simulacions dels episodis no han variat i continuen sent molt ràpides. La lentitud és causada per l'emmagatzemament dels diccionaris, que ocupen uns 400 MB cadascun. Com més gran és el diccionari, més temps es necessita per emmagatzemar-lo.

Quant al rendiment dels agents, de tant en tant es veu alguna decisió correcta, però, en general, no han après a jugar.

S'ha repetit l'experiment, utilitzant també la puntuació negativa, per comprovar si el problema és la recompensa utilitzada. A més, per resoldre el problema de la lentitud en l'emmagatzemament, els diccionaris s'emmagatzemen cada 100.000 partides.

Aquest experiment ha necessitat només dues hores per finalitzar, però els agents continuen sense saber jugar. Per aquest motiu, s'ha ampliat amb 500.000 episodis addicionals, continuant amb el valor d'ε tal com ha acabat l'experiment (a 0), de manera que l'actualització de la política per a l'ampliació ha estat utilitzant el mètode greedy. Els diccionaris s'han emmagatzemat només en finalitzar tots els episodis.

Aquesta ampliació ha necessitat 1 hora i 30 minuts entre la càrrega dels agents, l'entrenament i l'emmagatzemament dels diccionaris actualitzats, que pesen entre 600 MB i 800 MB cadascun. Els agents, però, no han mostrat cap millora significativa. En total s'han necessitat 3 hores i 30 minuts per entrenar dos agents amb 1.000.000 d'episodis.

Però, els agents continuen sense saber jugar. Només en alguna situació es veu alguna jugada bona, però podria ser cosa de l'atzar (amb tres cartes a la mà, hi ha el 33% de les probabilitats d'escollir l'acció correcta).

Segon experiment

Aquest segon experiment se centra a augmentar el nombre d'episodis per a l'entrenament dels agents en la variant “MC-OS” en la modalitat de dos jugadors per a la Brisca.

Després de finalitzar l'experiment anterior, s'ha investigat la causa per la qual els agents no escullen les accions correctes. S'ha detectat que la major part de les vegades es trien accions aleatòries perquè l'agent no ha visitat l'estat en qüestió durant l'entrenament.

S'ha utilitzat la variant “MC-OS” amb un valor d'exploració ε del 0,05, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001, amb un total de 5.000.000 d'episodis. Els diccionaris s'emmagatzemaren cada 1.000.000 d'episodis. De nou, s'enfronten dos agents entre si.

L'execució d'aquest experiment s'ha parat per falta de memòria RAM després de finalitzar 3.000.000 d'episodis (entre els 3 i els 4 milions). Els diccionaris han crescut tant que es necessita massa memòria RAM i l'entrenament no pot seguir. Tot i això, s'ha provat de continuar amb l'entrenament, i ja s'ha pogut observar que la memòria RAM utilitzada és molt elevada amb només la càrrega dels diccionaris a memòria. Ara, tan sols s'han pogut executar 1.000.000 d'episodis més, però l'equip també s'ha quedat sense memòria RAM en l'últim milió d'episodis.

Els diccionaris Q i parelles visitades ocupen uns 2,4 GB cadascun, mentre que els de la política i les accions ocupen 1,4 GB cadascun, sumant uns 15 GB en total (7,5 GB per agent). Aquests valors tan elevats són la causa que ocupin la major part de la memòria RAM disponible només carregar-los.

En total, considerant el temps d'emmagatzemament dels diccionaris, l'experiment per als 4.000.000 d'episodis completats ha durat 16 hores. A més, s'ha notat que l'emmagatzemament en episodis avançats és un procés que requereix més d'una hora.

Tot i que s'ha observat una millora en la presa de decisions dels agents en aquest experiment, les decisions aleatòries continuen provocant aquesta inconsistència en la presa de decisions.

Tercer experiment

En finalitzar l'experiment anterior, s'ha arribat a la conclusió que els agents no han pogut visitar tots els estats a causa del nombre limitat d'episodis executats, i augmentar-los no és viable a causa de l'ús excessiu de memòria RAM.

En aquest tercer experiment, s'utilitza un únic agent en la variant “MC-OS” en la modalitat de dos jugadors per a la Brisca, jugant contra si mateix. Això permet que l'agent pugui explorar el mateix nombre d'estats en la meitat d'episodis i de temps respecte a l'experiment anterior.

S'ha utilitzat un valor d'exploració ε de 0,1, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001. El valor de decreixement està pensat perquè el valor d'ε arribi a 0 després d'1.000.000 d'episodis, amb un total de 2.000.000 d'episodis. Els diccionaris s'emmagatzemen cada 1.000.000 d'episodis.

L'experiment ha finalitzat en 12 hores i no ha tingut cap problema amb la memòria RAM. Els diccionaris tenen un pes similar als de l'experiment anterior, demostrant que es pot replicar l'experiment amb menys episodis i menys temps.

Tal com s'esperava, els agents continuen prenent moltes decisions de manera aleatòria, ja que s'han explorat el mateix nombre d'estats que en l'experiment anterior. S'han visitat un total de 80.000.000 d'estats diferents considerant que no s'ha repetit la visita a cap estat, que hi ha 20 rondes per episodi i que l'agent aprèn del punt de vista tots dos jugadors. No obstant això, aquest nombre és inferior als 2^{40} estats diferents que es podrien considerar només amb la informació de les cartes vistes al llarg de la partida, un valor molt superior a l'hipotètic nombre d'estats visitats en aquest experiment.

Quart experiment

Després d'analitzar diverses partides de l'agent de l'experiment anterior, s'ha observat molta variabilitat en les seves decisions per a situacions similars, pel fet que es produeixen en moments o amb puntuacions dels jugadors diferents, que modifiquen l'identificador de l'estat. Per exemple, l'agent pot actuar de manera diferent per una mateixa configuració de cartes a la mà i jugades a la ronda segons si es tracta de la primera o de la segona ronda, o si el jugador 1 té dos punts en comptes de quatre.

En aquest quart experiment, s'ha eliminat la informació de les cartes vistes al llarg de la partida i la puntuació dels jugadors. L'objectiu és reduir el nombre total d'estats per tal que tots puguin ser visitats amb el mateix nombre d'episodis.

L'experiment continua utilitza la variant “MC-OS” amb un únic agent, un valor d'exploració ε de 0,1, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001, amb un total de 2.000.000 d'episodis. Els diccionaris s'emmagatzemen cada 1.000.000 d'episodis.

El temps necessari per finalitzar aquest experiment s'ha reduït a 10 hores, ja que els diccionaris són més petits i el procés d'emmagatzematge és inferior. Els diccionaris Q i parelles visitades ocupen 1 GB cadascun, mentre que el de la política i el de les accions ocupen uns 600 MB.

Amb menys informació per a l'estat del joc, l'identificador numèric que representa l'estat és més petit (té menys dígits), i, per tant, ocupa menys espai, reduint considerablement la mida dels diccionaris utilitzats.

No obstant això, encara hi ha massa estats i l'agent continua prenent moltes decisions de manera aleatòria. Es planteja la possibilitat de repetir l'experiment augmentant el nombre d'episodis, ja que la mida actual dels diccionaris hauria de permetre aquest augment, però la reducció de la informació de l'estat que s'ha realitzat pot afectar a les estratègies que els agents poden aprendre i desenvolupar, i s'ha decidit utilitzar un altre enfocament.

Cinquè experiment

En aquest cinquè experiment, s'ha entrenat un agent per a la variant “MC-MS” en la modalitat de dos jugadors per a la Brisca amb l'objectiu d'eliminar les decisions aleatòries de l'agent en situacions que no ha vist durant l'entrenament, però sí en situacions similars.

L'experiment s'ha realitzat amb un únic agent que juga contra si mateix, amb un valor d'exploració ε de 0,1, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001, amb un total de 2.000.000 d'episodis. Els diccionaris s'emmagatzemen cada 1.000.000 d'episodis.

El temps necessari per finalitzar l'experiment ha augmentat a 15 hores, principalment a causa de l'increment en la mida dels diccionaris. Aquesta vegada els diccionaris Q i parelles visitades ocupen 3 GB, i el de la política i les accions 2 GB. Això fa que es requereixi més temps per a l'emmagatzemament. Aquest augment és causat per la separació de l'estat en una tupla, ja que així es necessita més espai per a les claus dels diccionaris.

S'ha ampliat l'experiment amb 1.000.000 d'episodis addicionals, però l'equip s'ha quedat sense memòria RAM i no s'ha pogut finalitzar.

La implementació de l'abstracció de l'estat del joc utilitza múltiples abstraccions per identificar situacions de més similars a menys. Però, el procés és molt lent per culpa de la grandària dels diccionaris i s'ha limitat a només una abstracció que inclou la carta de triomf, les cartes jugades de la ronda i les cartes de la mà del jugador. Malgrat això, el temps de cerca encara és alt.

Després d'observar diverses partides de l'agent, s'ha comprovat que l'agent continua prenent decisions de manera aleatòria, però només entre tres i cinc vegades per partida. Això demostra que l'abstracció implementada funciona correctament.

Sisè experiment

Aquest sisè experiment s'ha centrat en la reducció de la mida dels diccionaris i l'ús de memòria RAM durant l'entrenament per permetre a l'agent completar 3.000.000 d'episodis i que disposi de més informació per aconseguir reduir el total de decisions que pren aleatòriament utilitzant la cerca de situacions similars.

Per aconseguir-ho, s'han unificat els diccionaris Q i parelles visitades que comparteixen, ja que comparteixen la mateixa clau (tupla “estat-acció”). Aquesta unificació ha permès reduir significativament l'espai físic necessari per als diccionaris, pel fet que la major part de la seva mida és la representació decimal de l'estat del joc.

A més, s'ha eliminat el diccionari d'accions, ja que l'agent pot determinar les accions que pot dur a terme consultant directament l'entorn, sense necessitat d'un diccionari específic.

L'experiment s'ha realitzat amb un únic agent de la variant “MC-MS” que juga contra si mateix, un valor d'exploració ε de 0,1, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001, amb un total de 3.000.000 d'episodis. Els diccionaris s'emmagatzemen cada 1.000.000 d'episodis.

Afortunadament, l'experiment ha finalitzat sense cap problema de memòria RAM, tot i que s'ha arribat al 98% de la capacitat en els moments finals de l'entrenament. El temps necessari per finalitzar l'experiment s'ha reduït a 12 hores, ja que només s'han emmagatzemat dos diccionaris, un de 6,7 GB que conté la unificació dels diccionaris Q i parelles visitades, i un de 3 GB per a la política.

L'emmagatzemament final dels diccionaris ha necessitat 2 hores i 15 minuts (1 hora i 30 minuts per al diccionari Q i les parelles visitades i 45 minuts per al de la política).

Després d'observar diverses partides de l'agent, s'ha verificat que només pren decisions aleatòries entre una i dues vegades per partida, per tant, pot prendre decisions davant de gairebé qualsevol situació. Tot i això, alguna de les seves decisions no és del tot correcte.

Entrenament final

L'experiment anterior ha establert el límit d'episodis per a l'entrenament amb aquesta tècnica. No s'ha aconseguit reduir més la mida dels diccionaris, i la memòria RAM utilitzada arriba al seu límit. Per tant, es procedeix a l'entrenament final dels agents per a la resta de jocs i modalitats.

L'entrenament s'ha realitzat amb un únic agent de la variant “MC-MS” que juga contra si mateix per a cada joc i modalitat, un valor d'exploració ε de 0,1, un valor de descompte γ d'1 i un decreixement d'ε del 0,0000001, amb un total de 3.000.000 d'episodis. Els diccionaris s'emmagatzemen cada 2.000.000 d'episodis. Cal destacar que els experiments anteriors s'han centrat en la modalitat de dos jugadors per a la Brisca, i es desconeix si l'execució de l'entrenament per a altres modalitats pot finalitzar prematurament perquè l'equip s'ha quedat

sense memòria RAM. D'altra banda, els diccionaris s'emmagatzemarien cada 3.000.000 d'episodis.

En total, s'han obtingut 12 agents per a la variant “MC-MS”, un per a cada joc i modalitat, sense aplicar cap regla opcional. La generació d'aquests agents ha requerit dues setmanes per completar-se. En particular, l'entrenament de cada agent per al joc de la Brisca ha durat 12 hores, mentre que per al joc del Tute 17 hores.

La mida total de tots els agents supera els 100 GB d'espai físic.

6.6. Aprenentatge genètic

La implementació realitzada per a l'aprenentatge genètic es divideix en quatre parts: les variants dels individus, l'avaluació i selecció dels millors individus, la generació de nova descendència i els experiments realitzats fins a obtenir els individus finals.

6.6.1. Variants dels individus

Variant “GA-XN”

Aquesta variant tracta d'aconseguir una xarxa neuronal que pugui aprendre a partir de l'encreuament i mutació dels gens dels millors individus, sense la necessitat d'entrenament amb un conjunt de dades inicial. D'aquesta manera, les estratègies dels individus no estaran limitades a les accions incloses en el conjunt de dades com passa en l'aprenentatge supervisat. No hi haurà exemples aleatoris, ni puntuació per ronda, ni partida guanyada / perduda, ni valor heurístic.

Un individu serà representat com una xarxa neuronal, amb els seus gens com els pesos i biaixos associats a les neurones. Les sortides de la xarxa neuronal correspondran a totes les possibles accions que l'individu podrà prendre en un moment donat de la partida; hi ha una sortida per a cada acció que es pot dur a terme.

A la Brisca hi haurà 41 sortides: les 40 primeres representen cadascuna de les cartes existents, i l'última indica l'acció d'intercanvi de la carta de triomf. Al Tute, s'afegeixen quatre sortides més que indiquen el cant en algun dels quatre colls.

D'aquesta manera, només caldrà realitzar una predicció per torn del jugador, en comptes d'una predicció per cada possible acció que l'individu pot realitzar i s'escollirà aquella acció amb més alta probabilitat de les accions possibles.

La població inicial estarà formada per xarxes neuronals creades amb pesos i biaixos aleatoris. S'ha aplanat cadascuna de les matrius de connexions de la xarxa neuronal en forma de vector per als pesos i biaixos, tal com es mostra a la figura 8, per poder dur a terme l'encreuament i mutacions dels individus per a la generació de noves poblacions.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rightarrow (1, 2, 3, 4, 5, 6, 7, 8, 9)$$

Figura 8: Exemple d'aplanada d'una matriu de connexions a vector per la variant “GA-XN”

Variant “GA-RL”

Aquesta variant té com a objectiu reduir la quantitat de temps que es necessita per a l'avaluació d'una generació amb la variant “GA-XN”, després de comprovar que la simulació de partides durant l'entrenament per reforç és prou ràpid.

Un individu serà un agent que aprendrà utilitzant la tècnica d'aprenentatge per reforç, i, els seus gens, els diccionaris necessaris per al seu entrenament.

Es comença amb una població inicial amb tots els diccionaris buits, i es procedeix a l'avaluació dels millors individus. Aquests agents aprendran mitjançant l'aprenentatge per reforç, utilitzant els diccionaris per a l'encreuament i mutació de les noves generacions.

6.6.2. Avaluació i selecció dels millors individus

A causa de la manca d'una funció d'avaluació clara per avaluar els individus individualment, es realitzaran emparellaments i simulacions de “N” partides contra altres individus de la població o contra altres models/agents entrenats amb altres tècniques d'aprenentatge, seleccionant aquells que hagin aconseguit més victòries i punts.

Els emparellaments entre individus de la mateixa població permeten que l'avaluació per a cada generació sigui més ràpida, ja que hi haurà menys emparellaments. No obstant això, l'avaluació no serà imparcial per a tots els individus, pel fet que alguns poden tenir emparellaments més fàcils que d'altres.

Per abordar aquest desavantatge, s'ha implementat la possibilitat de dur a terme una doble selecció dels millors individus. En la primera selecció, es triaran els “M·2” millors, i es realitzaran nous emparellaments entre ells per seleccionar finalment els “M” millors individus.

Els emparellaments amb altres models o agents entrenats amb altres tècniques d'aprenentatge permeten una avaluació imparcial per a tots els individus (excepte per l'atzar en el repartiment de les cartes), ja que tots s'enfronten al mateix rival. No obstant això, l'avaluació és més lenta, perquè implica que hi ha més emparellaments per generació i el temps de computació pot augmentar depenent del model/agent escollit.

S'ha implementat la tècnica de selecció d'elit (elite selection), en la qual els individus més aptes d'una generació són mantinguts en la següent. Així es garanteix que, si la nova descendència és menys apta, es podrà continuar amb els individus de la generació anterior (si aquests continuen sent els millors). Però, aquesta tècnica pot limitar l'exploració genètica, pel fet que hi ha menys descendència i menys variabilitat.

També s'ha implementat la possibilitat d'executar simulacions d'emparellaments en paral·lel, executant-se en fils diferents, reduint el temps de computació necessari i accelerant el procés d'avaluació dels individus.

6.6.3. Generació de nova descendència

Es trien dos individus de forma aleatòria d'entre els millors de la generació anterior i es genera un valor aleatori entre 0 i 1. Si aquest valor és inferior a la probabilitat d'encreuament, s'aplica algun dels operadors d'encreuament implementats per a cada variant. En cas contrari, els individus generats seran una còpia dels originals.

Per a la variant “GA-XN”, s'aplicarà aleatòriament un dels següents mètodes d'encreuament als vectors dels individus: [23, 26]

- Encreuament en un punt (one point crossover). Es divideix el vector dels pares en un punt de tall; la primera part del vector del pare 1 serà la primera del fill 1 i la segona del fill 2, i viceversa amb el pare 2.

  $\text{pare } 1 = (1,2,3,4,5,6)$, $\text{pare } 2 = (a,b,c,d,e,f)$, $\text{punt de tall} = 2$

  $\text{fill } 1 = (1,2,c,d,e,f)$

  $\text{fill } 2 = (a,b,3,4,5,6)$

  Figura 9: Exemple d'encreuament en un punt

- Encreuament en més d'un punt (multi point crossover). Se seleccionen diversos punts de tall per alternar les parts dels vectors dels pares als fills.

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(a,b,c,d,e,f),\ \text{punts de tall}=[2,3]$$
$$\text{fill } 1=(1,2,c,d,5,6)$$
$$\text{fill } 2=(a,b,3,4,e,f)$$

Figura 10: Exemple d'encreuament en més d'un punt

- Encreuament uniforme (uniform crossover). Cada gen és traspassat a un dels fills de forma aleatòria.

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(a,b,c,d,e,f)$$
$$\text{fill } 1=(1,2,c,4,5,f)$$
$$\text{fill } 2=(a,b,3,d,e,6)$$

Figura 11: Exemple d'encreuament uniforme

- Encreuament mitjà (average crossover) i encreuament pla (flat crossover). En el primer, es calcula el valor mitjà dels gens dels pares per generar un fill i, en el segon, per a cada gen es genera un valor aleatori que es troba entre els valors dels gens dels pares.

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(2,3,4,5,6,7)$$
$$\text{fill}=(1.5,2.5,3.5,4.5,5.5,6.5)$$

Figura 12: Exemple d'encreuament mitjà

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(2,3,4,5,6,7)$$
$$\text{fill}=(1.1,2.7,3.8,4.2,5.1,6.9)$$

Figura 13: Exemple d'encreuament pla

- Encreuament multivariant (multivariate crossover). Similar a l'encreuament en més d'un punt, però les parts dels vectors són traspassades als fills de manera aleatòria.

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(a,b,c,d,e,f),\ \text{punts de tall}=[2,3]$$
$$\text{fill } 1=(1,2,3,4,e,f)$$
$$\text{fill } 2=(a,b,c,d,5,6)$$

Figura 14: Exemple d'encreuament multivariant

- Encreuament aritmètic (arithmetic crossover). Es genera un valor aleatori entre 0 i 1 anomenat “alpha”. El primer fill hereta els gens resultants de la suma de multiplicar els gens del primer pare per “1 – alpha” i de multiplicar els del segon per “alpha”. El segon fill hereta els gens resultants de la suma de multiplicar els gens del segon pare per “1 – alpha” i de multiplicar els del primer per “alpha”.

$$\text{pare } 1=(1,2,3,4,5,6),\ \text{pare } 2=(2,3,4,5,6,7),\ \alpha=0.6$$
$$\text{fill } 1=(1-\alpha)\times\text{pare } 1+\alpha\times\text{pare } 2$$
$$\text{fill } 1=0.4\times\text{pare } 1+0.6\times\text{pare } 2$$
$$\text{fill } 1=(0.4,0.8,1.2,1.6,2,2.4)+(1.2,1.8,2.4,3,3.6,4.2)$$
$$\text{fill } 1=(1.6,2.6,3.6,4.6,5.6,6.6)$$

$$\text{fill } 2=\alpha\times\text{pare } 1+(1-\alpha)\times\text{pare } 2$$
$$\text{fill } 2=0.6\times\text{pare } 1+0.6\times\text{pare } 2$$
$$\text{fill } 2=(0.6,1.2,1.8,2.4,3,3.6)+(0.8,1.2,1.6,2,2.4,2.8)$$
$$\text{fill } 2=(1.4,2.4,3.4,4.4,5.4,6.4)$$

Figura 15: Exemple d'encreuament aritmètic

Per a la variant “GA-RL”, primer es procedeix a combinar els diccionaris dels pares. S'afegeixen totes les entrades que el pare 1 no té del pare 2, i viceversa. Però, si alguna clau és compartida pels dos pares, cadascun manté el valor del seu diccionari.

A continuació, s'aplica aleatòriament l'encreuament als parells “clau-valor” dels diccionaris amb algun dels mètodes següents: encreuament en un punt, encreuament en més d'un punt, encreuament uniforme o encreuament multivariant.

Un cop generats tots els individus, es procedeix amb la seva mutació. S'escull un valor aleatori entre 0 i 1 per a cada nou individu, i, si aquest és inferior a la probabilitat de mutació, s'aplica algun dels operadors de mutació implementats. En cas contrari, els gens de l'individu queden inalterats.

Per a la variant “GA-XN”, s'aplica aleatòriament un dels següents mètodes de mutació als vectors dels individus: [24]

- Mutació aleatòria parcial (partial shuffle mutation). S'escullen dos punts aleatoris i es barallen els gens que hi ha entre ells.

$$\text{individu} = (1,2,3,4,5,6),\ \text{punts aleatoris} = [1,4]$$
$$\text{individu} = (1,3,5,2,4,6)$$

Figura 16: Exemple de mutació aleatòria parcial

- Mutació d'inversió (inversion mutation). Es trien dos punts aleatoris i s'inverteix l'ordre dels gens entre aquests punts.

$$\text{individu} = (1,2,3,4,5,6),\ \text{punts aleatoris} = [1,4]$$
$$\text{individu} = (1,5,4,3,2,6)$$

Figura 17: Exemple de mutació d'inversió

- Mutació de desplaçament (displacement mutation). S'escullen dos punts aleatoris i traspassa el segment entre aquests punts a una nova posició.

$$\text{individu} = (1,2,3,4,5,6,7,8,9),\ \text{punts aleatoris} = [4,7],\ \text{nova posició} = 2$$
$$\text{individu} = (1,2,5,6,7,3,4,8,9)$$

Figura 18: Exemple de mutació de desplaçament

- Mutació d'inversió de desplaçament (displacement inversion mutation). Similar a la mutació de desplaçament, però capgirant l'ordre dels gens abans de traspassar-los.

$$\text{individu} = (1,2,3,4,5,6,7,8,9),\ \text{punts aleatoris} = [4,7],\ \text{nova posició} = 2$$
$$\text{individu} = (1,2,7,6,5,3,4,8,9)$$

Figura 19: Exemple de mutació d'inversió de desplaçament

Per a la variant “GA-RL”, s'utilitza només el següent operador de mutació:

- Mutació de reinici aleatori (random resetting mutation). Se seleccionen elements aleatoris del diccionari de la política de l'individu i s'actualitza el seu valor amb una nova acció associada a aquest estat de manera aleatòria.

Les probabilitats d'encreuament i mutació poden ser fixades per a tot l'entrenament o ajustades al llarg de les generacions. S'han implementat diverses directives de control per gestionar aquestes probabilitats. [25]

- Dynamic Increasing of Low Mutation / Decreasing of High Crossover (ILM/DHC). Comença l'entrenament amb una alta probabilitat d'encreuament i una baixa probabilitat de mutació. Amb el pas de les generacions, la probabilitat d'encreuament disminueix gradualment fins a arribar al 0%, mentre que la de mutació incrementa fins a arribar al 100%. Aquesta directiva dona millors resultats quan es disposa d'una població petita. El càlcul de les probabilitats es pot veure a la fórmula 20:

$$\text{crossover\_rate} = 1 - \text{mutation\_rate}$$
$$\text{mutation\_rate} = \frac{\text{generacio\_actual}}{\text{generacions\_totals}}$$

Fórmula 20: càlcul de probabilitats d'encreuament i mutació (ILM/DHC)

- Dynamic Decreasing of High Mutation Rate / Increasing of Low Crossover Rate (DHM / ILC). Comença amb una baixa probabilitat d'encreuament i una alta probabilitat de mutació. Amb el pas de les generacions, les probabilitats s'inverteixen. Aquesta directiva dona millors resultats quan es disposa d'una població gran. El càlcul de les probabilitats es pot veure a la fórmula 21:

$$\text{mutation\_rate} = 1 - \text{crossover\_rate}$$
$$\text{crossover\_rate} = \frac{\text{generacio\_actual}}{\text{generacions\_totals}}$$

Fórmula 21: càlcul de probabilitats d'encreuament i mutació (DHM/ILC)

- Fixed 50% for Mutation and Crossover Rates (FFMCR). Es mantenen les probabilitats d'encreuament i mutació al 50% durant tot l'entrenament.
- Parameter Tuning Method (PTM). Es mantenen les probabilitats d'encreuament al 93% i la de mutació al 3% durant tot l'entrenament.
- Personalitzable. Permet personalitzar les probabilitats d'encreuament i mutació.

6.6.4. Experiments i entrenament dels individus

S'han dut a terme diversos experiments per provar diferents configuracions de paràmetres com el total d'individus per generació, el total de generacions, els diferents mètodes implementats per a la selecció dels millors individus i la utilització de les diferents variants que es detallen a continuació.

Els individus només poden ser entrenats per a una combinació de regles opcionals específica. La idea és obtenir uns individus capaços de jugar als jocs en la forma més bàsica, sense cap regla opcional. A continuació, els millors individus de l'última generació s'utilitzaran com a població inicial per a un nou entrenament amb altres regles opcionals.

Tots els experiments detallats s'han executat sense cap regla opcional i s'han mesurat en dies d'entrenament, però no són ininterromputs (24 hores seguides), sinó que cada dia equival a entre 8 i 12 hores.

Primer experiment

Aquest primer experiment serveix com a punt de contacte amb la variant “GA-XN” i se centra a replicar les xarxes neuronals obtingudes amb la tècnica d'aprenentatge supervisat, però utilitzant aprenentatge genètic.

S'ha utilitzat la variant “GA-XN” per a la modalitat de dos jugadors de la Brisca, generant una població de 16 xarxes neuronals de dues capes amb 75 i 50 neurones, utilitzant totes les entrades de la xarxa en format binari. En el moment de la realització d'aquest experiment, encara no s'havia realitzat la reducció de la dimensionalitat de les entrades a l'aprenentatge supervisat.

Els individus són avaluats contra el model de dues capes de 75 i 50 neurones de la variant “SL-PH”, i s'han escollit els 4 millors per a generar la nova descendència. La directiva de control utilitzada és “ILM / DHC” i s'han simulat 10 partides per cada emparellament durant 100 generacions.

L'experiment ha requerit 7 dies per finalitzar. S'han simulat 160 partides per cada generació, sumant un total de 16.000 partides. Si es té en compte que cal realitzar una predicció per cada possible acció amb la a la variant “SL-PH” (3 cartes a la mà, 3 prediccions) i una per a la variant “GA-XN” i que hi ha un total de 20 rondes per partida, s'han realitzat un total d'1.280.000 prediccions.

$$(1+3\ prediccions\ per\ ronda)\times 20\ rondes\times 12000\ partides = 1280000\ prediccions$$

Fórmula 22: Nombre de prediccions per 100 generacions. 16 individus vs agent supervisat

Tot i observar un parell de partides dels millors individus de l'última generació, cap d'ells ha mostrat que sap jugar correctament. És evident que es requereix moltes més generacions per obtenir individus que aprenguin a jugar mitjanament bé. No obstant això, el temps necessari per a la finalització de l'entrenament limita el nombre de generacions que es poden executar.

Segon experiment

En aquest segon experiment s'intenta millorar l'eficiència en l'avaluació de les generacions de la variant “GA-XN” eliminant el rival de la variant “SL-XN”. S'utilitzarà els mateixos individus de la població per realitzar els emparellaments, reduint el temps necessari per a cada avaluació.

L'experiment se centra en la modalitat de dos jugadors de la Brisca, amb l'ús de 32 individus (mateix nombre d'emparellaments) i la selecció dels 8 millors per a la generació de la nova descendència. S'aplica la directiva de control “ILM / DHC” i se simulen 16 partides per emparellament (256 partides per generació) durant 100 generacions.

Tot i que s'han simulat més partides al final de l'experiment, només ha requerit dos dies per finalitzar. Aquesta diferència de temps s'atribueix al fet que les prediccions dels individus de la variant “GA-XN” són més ràpides que les de la variant “SL-XN”, pel fet que té 40 inputs binaris menys. Això implica menys càlculs per a la predicció de la millor acció en la propagació cap endavant.

S'ha repetit l'experiment reduint el nombre d'individus a 16, seleccionant els 4 millors de cada generació i reduint el nombre de partides per emparellament a 10 per equiparar les condicions amb les del primer experiment.

Aquesta configuració ha permès completar l'experiment en només un dia. No obstant això, s'ha arribat a la conclusió que la reducció del nombre de partides per emparellament incrementa el paper de l'atzar (a major nombre de partides per emparellament, els resultats són més fiables). A més, l'avaluació dels individus ja no és imparcial, ja que no competeixen contra el mateix rival.

Aquestes conclusions han portat a la implementació del mètode de doble selecció dels millors individus. No obstant això, s'ha observat que aquest mètode incrementa el nombre de partides i el temps d'execució (dos dies amb la mateixa configuració d'individus i nombre de partides), i, per tant, s'ha abandonat la idea.

Tercer experiment

En aquest tercer experiment s'ha implementat l'execució d'emparellaments en paral·lel i se centra en la comparació dels temps necessaris per a l'entrenament de quatre generacions amb diferents configuracions de nombre d'individus, nombre d'execucions en paral·lel i activació de la GPU per a la variant “GA-XN”.

La taula 13 mostra els resultats obtinguts d'aquest experiment, on se simulen 10 partides per emparellament per a la modalitat de dos jugadors de la Brisca.

Individus	Partides per emparellament	Fils	Ús de GPU	Temps d'entrenament
16	10	1	No	20m
16	10	8	No	10m
64	10	1	No	53m
64	10	16	No	30m
64	10	32	No	30m
64	10	32	Si	28m

Taula 13: Comparativa de temps d'entrenament de quatre generacions executant simulacions en paral·lel

Es nota una reducció considerable del temps necessari quan s'utilitza vuit fils en comptes d'un amb 16 individus, on el temps es redueix a la meitat. No obstant això, a mesura que s'augmenta el nombre de fils, aquesta reducció en el temps disminueix, arribant a un punt on ja no es produeix cap millora (el temps d'execució per a 64 individus amb 16 fils és igual que amb 32 fils). Utilitzar 8 fils redueix el temps d'entrenament en un 50%, mentre que amb 16 un 43%.

Per tant, sembla que hi ha un límit en el nombre de fils que es poden utilitzar de forma simultània. És probable que el processador no pugui atendre els càlculs requerits per cada fil de manera simultània i hagi de fer “torns” per processar-los. En conseqüència, tot i que els emparellaments s'executen simultàniament, no tots són atesos alhora.

Pel que fa a l'activació de la GPU, no s'observa una millora en el temps d'entrenament, ja que només es realitzen càlculs de propagació cap endavant i no de retropropagació, que són càlculs complexos on sí que es nota aquesta activació.

Quart experiment

El quart experiment se centra a aplicar la reducció de la dimensionalitat implementada en l'aprenentatge supervisat per verificar si es pot reduir el temps de predicció de la variant “GA-XN”.

Inicialment, es fa un primer experiment per a la modalitat de dos jugadors de la Brisca on s'eliminen les cartes conegudes dels rivals i s'utilitza el mètode de la selecció d'elit, tot i que no aporta res a la reducció del temps de predicció.

S'utilitzen dues capes per a cada individu, però s'ha reduït el nombre de neurones a 50 i 25. La població consta de 32 individus i se seleccionen els 8 millors per a la generació de la nova descendència. Es manté la directiva de control “ILM / DHC” i se simulen 10 partides per a cada emparellament durant 100 generacions. Com que hi ha 16 emparellaments, s'utilitzen 16 fils diferents, el mateix processador limitarà les peticions.

Aquest experiment ha finalitzat en un dia, equiparant el temps del segon experiment, però utilitzant el doble d'individus per població. No obstant això, el temps encara és molt elevat per a entrenaments amb moltes generacions. Per exemple, per entrenar 100.000 generacions es necessitarien 208 dies d'entrenament ininterromput.

S'ha repetit l'experiment amb 16 individus, utilitzant 8 fils i seleccionant només els 4 millors de cada generació. En aquest cas, l'experiment ha finalitzat en 5 hores, aproximadament la meitat del segon experiment amb la mateixa quantitat d'individus i partides. Així, considerant que amb 8 fils es redueix un 50% el temps d'entrenament, la reducció de la dimensionalitat no aporta gaire reducció de temps.

Cinquè experiment

En aquest cinquè experiment s'ha aplicat la reducció completa de la dimensionalitat de les dades d'entrada, utilitzant la mateixa representació normalitzada vista en l'entrenament supervisat, per comparar els temps d'entrenament amb els experiments anteriors per a la variant “GA-XN”. A més, s'ha realitzat un entrenament de 1.000 generacions per comprovar si es poden observar millores en la manera de jugar dels individus.

Inicialment, s'han repetit els experiments anteriors per a la modalitat de dos jugadors de la Brisca, utilitzant poblacions de 32 i 16 individus. No obstant això, els temps d'entrenament han estat similars i no s'ha pogut apreciar una millora significativa.

Per a l'experiment de les 1.000 generacions s'ha continuat utilitzant dues capes per a cada individu de 50 i 25 neurones. La població consta de 32 individus, s'executen 16 fils en paral·lel i se seleccionaran els 6 millors per a la generació de la nova descendència. La directiva de control utilitzada és “ILM / DHC” i se simulen 10 partides per a cada emparellament.

Aquest experiment s'ha finalitzat en cinc dies, però no s'ha observat cap millora en el rendiment dels individus. Es pot apreciar alguna jugada bona cada dues o tres partides, però no es pot atribuir a una millora dels individus, sinó més aviat a l'atzar.

Arribats a aquest punt, es decideix abandonar aquesta tècnica, almenys fins a obtenir un agent entrenat per reforç (en aquest moment encara no s'havia implementat l'algoritme per reforç).

Sisè experiment

En aquest sisè experiment es descarta la idea d'utilitzar un agent entrenat per reforç com a rival dels individus en els emparellaments, ja que la presa de decisions no és tan ràpida com s'esperava. Ho és quan es disposa de l'estat exacte al diccionari de la seva política, però no quan ha de cercar situacions similars amb l'abstracció de l'estat.

No obstant això, la implementació de la tècnica d'aprenentatge per reforç ha donat lloc a la idea de la variant “GA-RL”, que substitueix la representació dels individus de xarxes neuronals per agents entrenats per reforç. D'aquesta manera, s'eliminen les prediccions lentes de la variant “GA-XN” i s'accelera tot el procés d'entrenament, ja que els agents per reforç són ràpids en la tria d'accions durant l'entrenament, aconseguint simular 1.000.000 de partides en unes 4 hores.

Per comprovar l'efectivitat d'aquesta nova variant, s'ha fet un primer experiment de 4 generacions per a la modalitat de dos jugadors de la Brisca. S'han utilitzat 200 individus i s'han escollit els 20 millors per a la generació de la nova descendència. La directiva de control utilitzada és “ILM / DHC” i s'han simulat 10 partides.

Pel que fa a la configuració de l'entrenament per reforç, s'ha utilitzat un valor d'exploració (ε) del 0,05 amb un decreixement del 0,0000001 i un valor de descompte (γ) d'1.

L'experiment ha estat un èxit i ha necessitat només 3 minuts per finalitzar. Extrapolant aquest resultat, 10.000 generacions serien, aproximadament, 5 dies d'entrenament, per la qual cosa, es decideix iniciar l'experiment amb aquesta mateixa configuració per 10.000 generacions.

No obstant això, lluny de les expectatives, després de 10 hores d'entrenament només s'han completat 25 generacions. El principal problema d'aquesta variant és l'encreuament i mutació dels diccionaris dels agents. A mesura que avancen les generacions, els diccionaris augmenten la seva mida i, com més gran són, més temps es necessita per a l'encreuament i mutació.

S'ha observat que es necessiten aproximadament dues hores per a l'encreuament i la mutació dels 200 individus per a la generació 25. A més, com més creixen els diccionaris, més temps es necessita per a la seva càrrega a memòria, endarrerint també l'inici dels emparellaments.

Encara que se solucionés el problema del temps d'encreuament i mutació, i es reduís el temps de càrrega dels agents, hi hauria el problema de la limitació per l'ús excessiu de la memòria RAM. En poques generacions ja no es podrien executar els emparellaments en paral·lel, ja que no hi hauria prou memòria RAM per carregar tots els individus alhora.

Finalment, es decideix donar per finalitzat l'entrenament genètic, disposant únicament de 6 individus de la variant “GA-XN”, els millors de l'última generació, entrenats amb 1.000 generacions per a la modalitat de dos jugadors per a la Brisca, sense utilitzar cap regla opcional.

7. Resultats

7.1. Aprenentatge supervisat

7.1.1. Models entrenats

S'han obtingut 24 models diferents de la variant “SL-PH” que poden jugar amb qualsevol combinació de regles. Hi ha dos models per a cada modalitat (Brisca, Tute i Tute amb només l'obligació d'assistir) per a dos, tres, quatre jugadors i equips.

La diferència entre els models d'una mateixa modalitat és l'arquitectura utilitzada de la xarxa neuronal utilitzada. La meitat dels models utilitzen dues capes amb 50 i 25 neurones, i l'altra meitat tres capes amb “inputs/2”, “inputs/4” i “inputs/6” neurones respectivament.

7.1.2. Temps d'entrenament, temps de càrrega i ús i mida dels models

Els temps d'entrenament varien segons la quantitat de jugadors, ja que hi ha una regla opcional que només s'aplica per a dos jugadors, cosa que fa que el conjunt de dades contingui més partides.

La taula 14 mostra un resum dels temps necessaris per a l'entrenament dels models. Es detallen el total de partides utilitzades per a l'entrenament de cada modalitat (tp), el temps de generació del conjunt de dades (tg), el temps de tractament del conjunt de dades (tt) i el temps d'entrenament del model (te).

Modalitat		tp	tg	tt	te (2 capes)	te (3 capes)
Brisca	**2j**	128000	19m	9m	1h 25m	1h 21m
	3j	64000	10m	6m	37m	39m
	4j	64000	11m	6m	38m	42m
	equips	64000	12m	6m	38m	42m
Tute	**2j**	128000	28m	11m	1h 21m	1h 27m
	3j	64000	14m	7m	41m	43m
	4j	64000	15m	7m	42m	45m
	equips	64000	15m	7m	42m	46m
Tute només assistir	**2j**	128000	30m	10m	1h 23m	1h 27m
	3j	64000	14m	7m	41m	43m
	4j	64000	16m	7m	42m	45m
	equips	64000	16m	7m	42m	45m

Taula 14: Temps de generació, tractament dels conjunts, temps d'entrenament dels models supervisats finals

Els models tenen una mida d'entre 80 i 110 KB, i els temps de càrrega són pràcticament inexistents. El poc pes dels models permet una càrrega instantània, de manera que no cal esperar per poder utilitzar-los.

D'altra banda, la predicció no és automàtica, ja que cal calcular amb les operacions de propagació cap endavant de la xarxa neuronal, i per escollir la millor acció en un torn, cal realitzar tantes prediccions com accions es poden dur a terme.

7.1.3. Recursos necessaris

L'entrenament utilitza un 40% de memòria RAM, un 35% de CPU i un 35% de GPU en cas que estigui activada. Si no ho està, l'ús de CPU no varia, però sí que augmenta el temps d'entrenament.

En canvi, la utilització dels models per a prediccions només necessita un 20% de memòria RAM i un 18% de CPU.

7.1.4. Gràfiques i mètriques

S'han emmagatzemat els resultats de les mètriques exactitud, precisió i sensibilitat de la validació de cada entrenament que s'ha realitzat. La figura 23 mostra una gràfica amb aquestes mètriques per a tots els entrenaments realitzats.

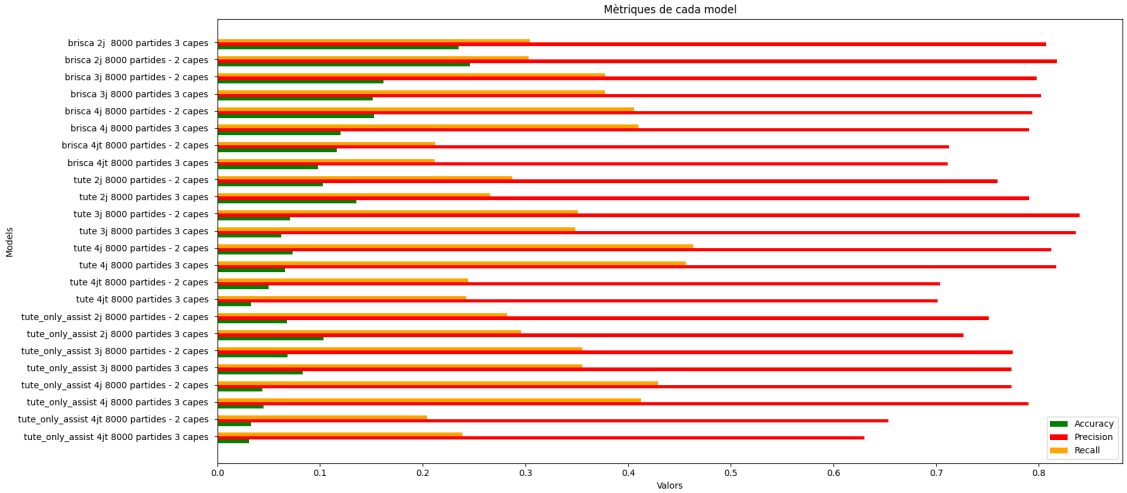

Figura 23: Mètriques de la validació dels models supervisats

La mètrica exactitud mesura la proporció d'exemples classificats correctament sobre el total d'exemples avaluats. Per a models amb múltiples sortides, es fa el càlcul per a cada etiqueta i es calcula la mitjana global.

$$exactitud = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}\right)$$, on “i” representa cadascuna de les etiquetes.

Fórmula 24: Mètrica exactitud

La mètrica sensibilitat mesura la capacitat del model per identificar correctament els exemples positius d'una classe sobre el total d'exemples d'aquesta classe. Igual que amb l'exactitud, es calcula la sensibilitat per a cada etiqueta i es calcula la mitjana global.

$$sensibilitat = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{TP_i}{TP_i + FN_i}\right)$$, on “i” representa cadascuna de les etiquetes.

Figura 25: Mètrica sensibilitat

La mètrica precisió mesura la proporció d'encerts de cada classe sobre el total de prediccions realitzades per aquella classe. De la mateixa manera que amb les altres mètriques, es calcula la precisió per a cada etiqueta i es calcula la mitjana global.

$$precisió = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{TP_i}{TP_i + FP_i}\right)$$, on “i” representa cadascuna de les etiquetes.

Figura 26: Mètrica precisió

A les gràfiques es pot veure que els models tenen una precisió elevada (entre el 0,6 i el 0,8), però la seva sensibilitat i la seva exactitud són baixes (entre 0,3 i 0,5 per la sensibilitat i entre

0,05 i 0,3 per l'exactitud). A més, no es nota gaire diferència entre les mètriques dels models de dues i tres capes.

La baixa exactitud i sensibilitat indica que només una proporció molt baixa de les prediccions coincideixen exactament amb el valor de la suma de la puntuació de la ronda i el valor de l'heurístic. Aquest resultat té molt sentit perquè, per exemple, en el cas de dos jugadors per a la Brisca, si un jugador comença una ronda, no se sap quines cartes tindrà el rival i el valor de la sortida pot prendre molts valors. En canvi, quan el jugador finalitza la ronda, té la certesa de la puntuació que s'emportarà.

En augmentar el nombre de jugadors, disminueix el nombre de situacions en què el jugador serà l'últim de la ronda i tindrà la certesa de la puntuació final, fet que fa disminuir l'encert en les prediccions. Aquest decreixement de l'exactitud en augmentar el nombre de jugadors també és present en el joc del Tute. No obstant això, l'encert de les prediccions en el Tute és més baix en general que a la Brisca. Això es deu al fet que en el Tute, un jugador pot tenir fins a vuit cartes a la seva mà, i, per tant, hi ha més incertesa sobre les cartes que jugarà el rival.

L'alt encert en la precisió indica que el model és capaç de predir correctament aquelles situacions que són molt clares (en principi les situacions en què el jugador és l'últim de la ronda).

Com es pot apreciar, aquestes mètriques no ajuden a diferenciar si un model ha après a jugar als jocs o no. Per aquest motiu, no s'han tingut en compte a l'hora d'avaluar els models.

7.2. Aprenentatge per reforç

7.2.1. Agents entrenats

S'han obtingut 12 agents per a la variant “MC-MS”, un per a cada joc i modalitat, entrenats específicament sense utilitzar cap regla opcional. No obstant això, poden jugar amb qualsevol combinació gràcies a l'abstracció que s'ha desenvolupat de l'estat del joc.

7.2.2. Temps d'entrenament, temps de càrrega i ús i mida dels models

En total, s'han necessitat 10 dies per entrenar els 12 agents. Els agents entrenats per a la Brisca han necessitat 12 hores per finalitzar un entrenament amb 3.000.000 d'episodis, mentre que per al Tute han necessitat 17 hores.

La mida de cada agent és d'entre 10 i 11 GB entre els dos diccionaris. Això fa que el temps de càrrega de l'agent sigui molt elevat (entre 2 i 3 minuts).

Les prediccions durant l'entrenament són molt ràpides, ja que sempre s'escull una acció de la seva política o una acció aleatòria si no s'ha visitat aquella situació específica, i no es genera cap abstracció de l'estat. No obstant això, el seu ús és molt lent si s'ha de cercar una situació similar per escollir una acció.

Per exemple, si es vol utilitzar algun dels agents per a una altra combinació de regles opcionals, totes les prediccions necessitaran cercar situacions similars, ja que la seva política no disposarà de cap entrada per aquesta combinació de regles opcionals.

7.2.3. Recursos necessaris

Durant l'entrenament dels agents, la memòria RAM necessària augmenta a mesura que els diccionaris creixen. En les etapes inicials, l'ús de memòria és pràcticament nul, però augmenta fins a gairebé el 100% en entrenaments de 3.000.000 d'episodis. Per la seva banda, l'ús de CPU és estable i requereix un 20%.

En canvi, la utilització de l'agent no necessita utilitzar la CPU, però l'ús de memòria RAM és molt elevat, ja que cal carregar el diccionari de la política de l'agent a memòria (no cal el diccionari Q i parelles visitades). S'ha aconseguit carregar a memòria dues polítiques alhora, però, en carregar una tercera, l'equip es queda sense memòria RAM.

7.3. Aprenentatge genètic

7.3.1. Individus entrenats

S'han obtingut 6 individus, amb dues capes de 50 i 25 neurones, per a la variant “GA-XN” en l'última generació de l'entrenament per a la modalitat de dos jugadors de la Brisca, sense aplicar cap regla opcional.

7.3.2. Temps d'entrenament, temps de càrrega i ús i mida dels models

L'entrenament de cada generació requereix aproximadament 5 minuts, però aquest temps augmenta depenent del nombre d'individus per població, l'ús d'un model o un agent com a rival en comptes dels mateixos individus o l'increment del nombre de partides per emparellament. En total s'han necessitat 5 dies d'entrenament per finalitzar les 1.000 generacions.

Els individus tenen una mida d'aproximadament 49 KB, fet que permet una càrrega a memòria instantània sense esperes.

Pel que fa a les prediccions, tot i que no són automàtiques, només cal una predicció per cada torn, independentment del nombre d'accions disponibles, cosa que redueix considerablement el temps necessari per a cada predicció.

7.3.3. Recursos necessaris

Durant l'entrenament dels individus, es requereix un 40% de memòria RAM, un 20% de CPU. En cas que s'utilitzi la GPU, aquesta utilitza un 15% de la seva capacitat. L'execució de múltiples emparellaments simultàniament incrementa l'ús de recursos en comparació amb l'aprenentatge supervisat, però només en un 20% de memòria RAM utilitzant menys percentatge de GPU.

En canvi, la utilització de l'individu només necessita un 20% de memòria RAM i un 20% de CPU, similar a l'ús dels models d'aprenentatge supervisat.

7.3.4. Gràfiques

La figura 27 mostra una gràfica que detalla la progressió de victòries i punts al llarg de les 1.000 generacions, destacant el millor individu cada 50 generacions.

Figura 27: Aprenentage genètic - Victòries i puntuació del millor individu cada 50 generacions

No s'observa cap millora en els models amb el pas de les generacions. En cada generació hi ha individus que aconsegueixen guanyar 8 de les 10 partides simulades per emparellament, i, a vegades, en destaca algun amb 9 victòries.

És important destacar que en aquestes primeres versions dels individus, les decisions es prenen de manera aleatòria, de manera que el nombre de victòries pot atribuir-se a l'atzar en el repartiment de les cartes i a l'aleatorietat dels emparellaments.

7.4. Rendiment i comparació dels agents

Per poder avaluar el rendiment dels diferents models, agents i individus obtinguts, s'han simulat 60 partides entre ells per a les diferents modalitats dels jocs. Aquest nombre de partides és suficient per minimitzar l'impacte de l'atzar en els resultats i alhora permet executar totes les simulacions en un temps raonable, que ha necessitat 5 dies per finalitzar-les.

En la modalitat de dos jugadors, s'han realitzat emparellaments perquè tots juguin contra tots sense aplicar cap regla opcional, ja que els individus i els agents han estat entrenats sense utilitzar cap regla opcional.

D'altra banda, els models supervisats tenen la capacitat de jugar amb qualsevol combinació de regles opcionals. Per tant, s'han simulat emparellaments entre ells per a cada combinació possible de regles. El codi “0000” indica que no s'ha aplicat cap regla opcional, mentre que “1100” indica que s'han aplicat les dues primeres (intercanvi de carta, 10 d'últimes, mà negra, caça del 3), seguint aquest ordre.

En les modalitats de quatre jugadors i per equips, només es disposa de 2 models i 1 agent. En la modalitat individual de quatre jugadors s'ha inclòs un jugador que pren decisions de manera aleatòria. Per a la modalitat per equips, cada equip està format per un mateix model o agent, de manera que el seu company és ell mateix.

Per als casos de tres i quatre jugadors, s'han repetit les simulacions variant la posició dels jugadors a la taula. Això es fa per tenir uns resultats imparcials, ja que jugar abans o després d'un jugador expert o inexpert pot afectar els resultats obtinguts.

7.4.1. Brisca

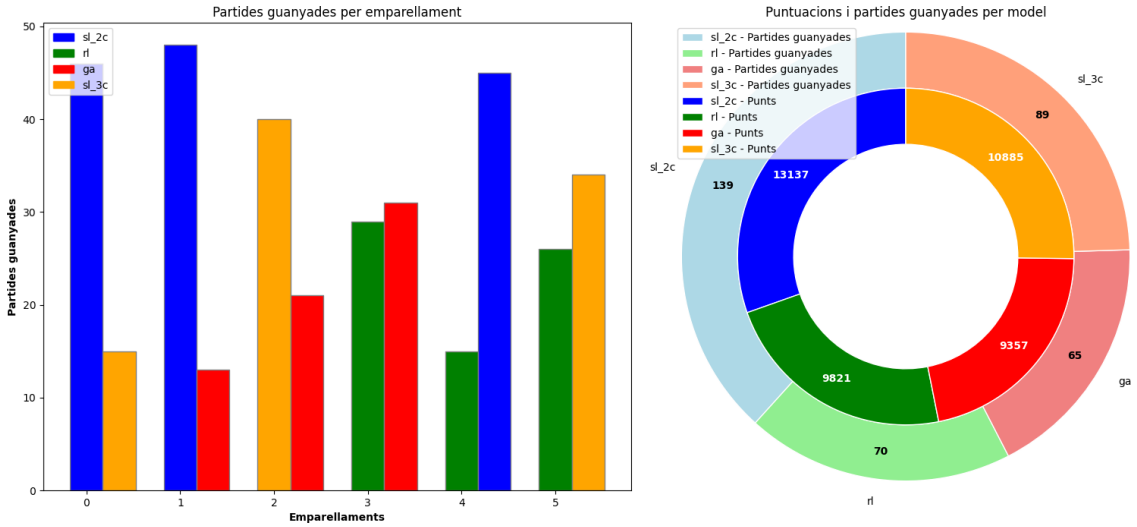

Figura 28: Resultats Brisca - dos jugadors

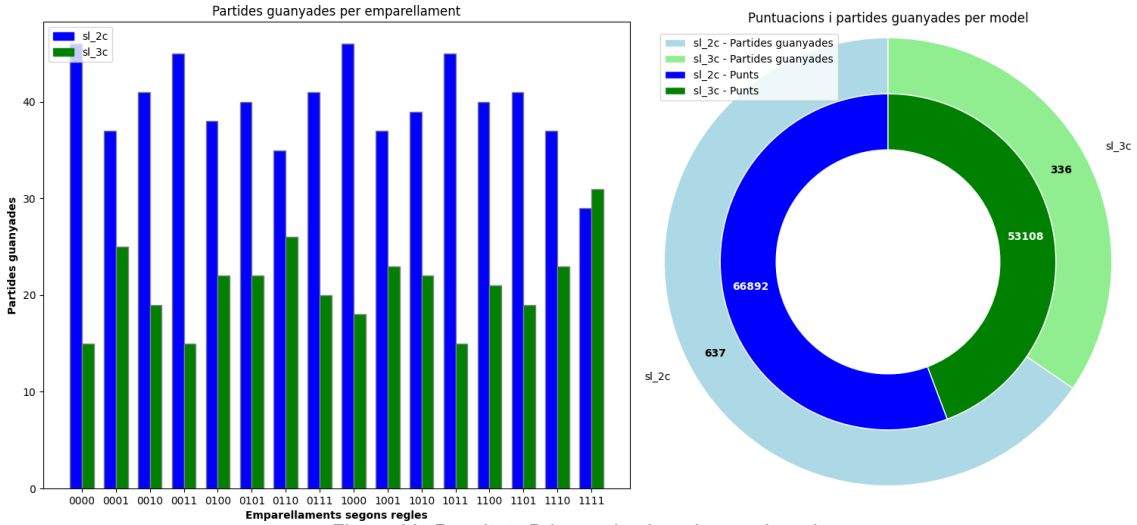

Figura 29: Resultats Brisca - dos jugadors amb regles

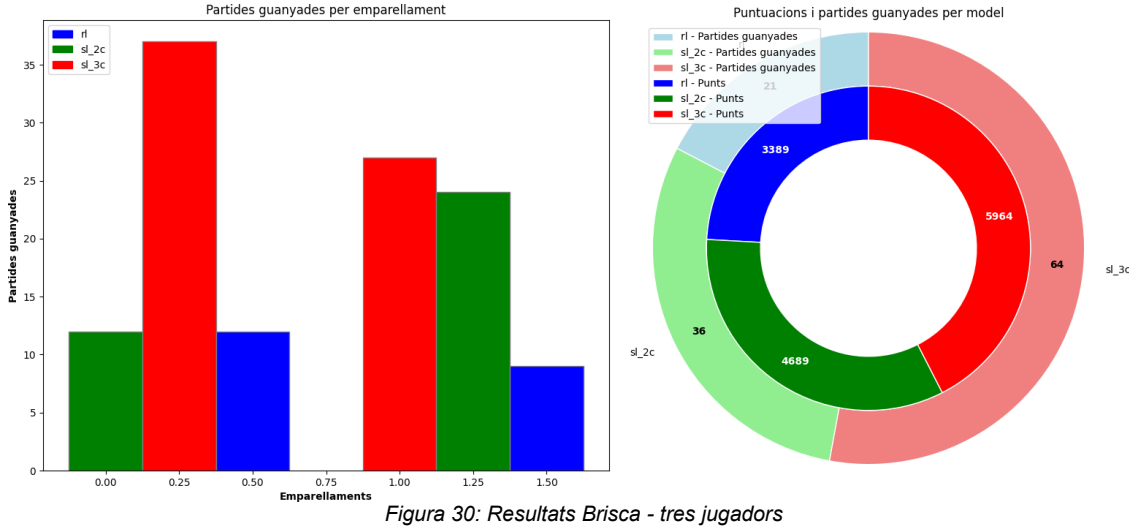

Figura 30: Resultats Brisca - tres jugadors

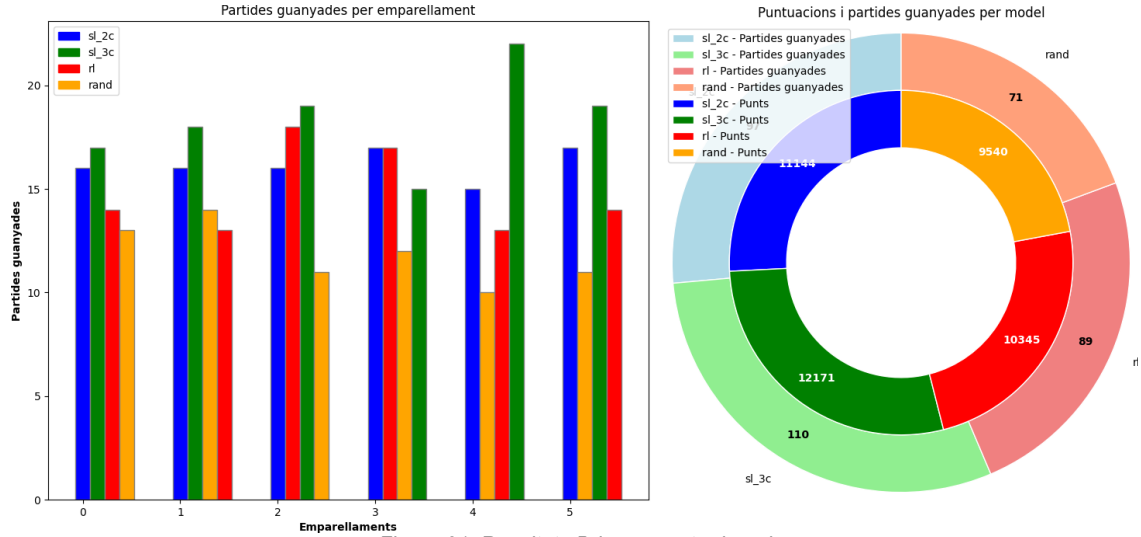

Figura 31: Resultats Brisca - quatre jugadors

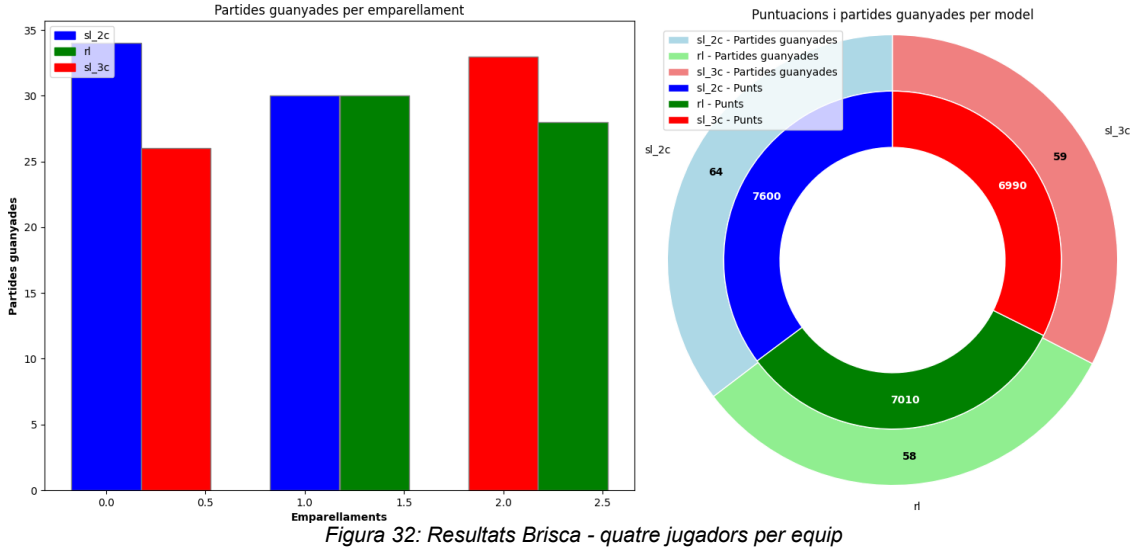

Figura 32: Resultats Brisca - quatre jugadors per equip

7.4.2. Tute

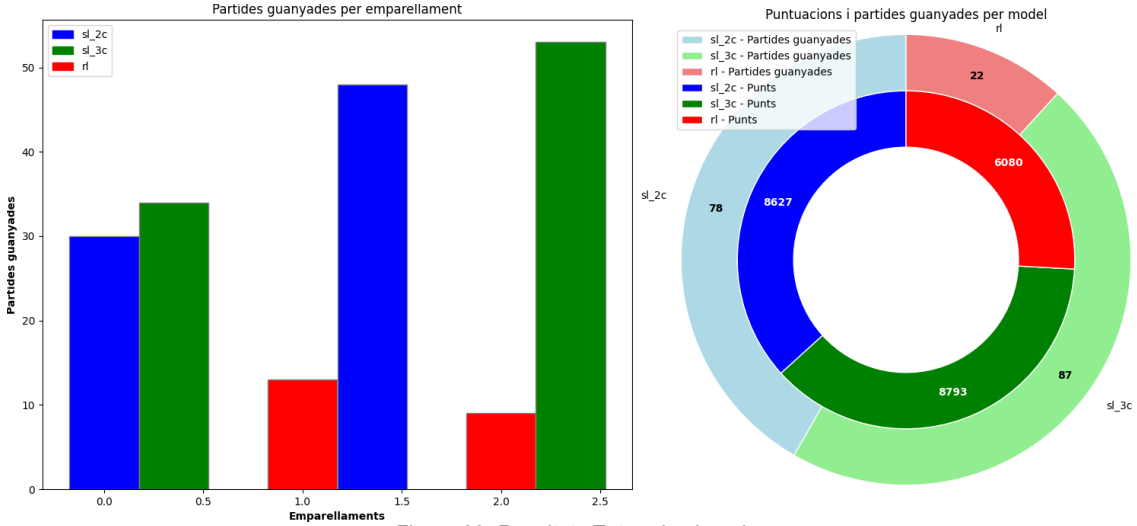

Figura 33: Resultats Tute - dos jugadors

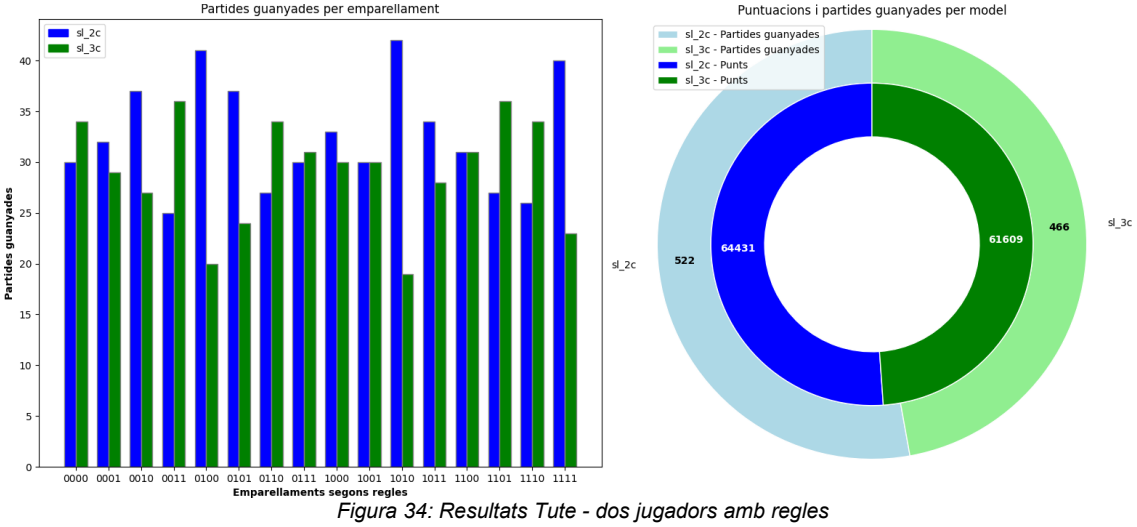

Figura 34: Resultats Tute - dos jugadors amb regles

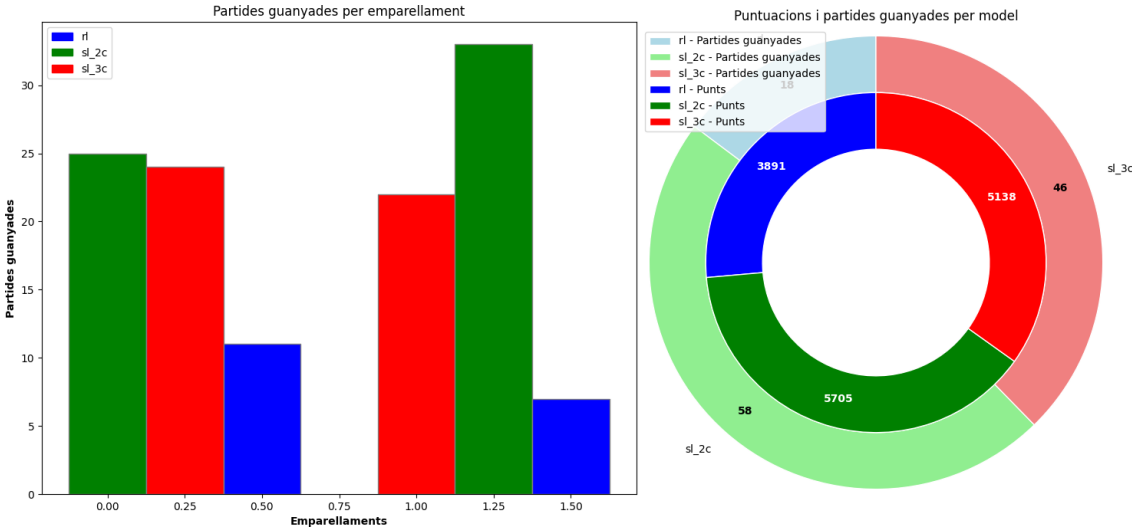

Figura 35: Resultats Tute - tres jugadors

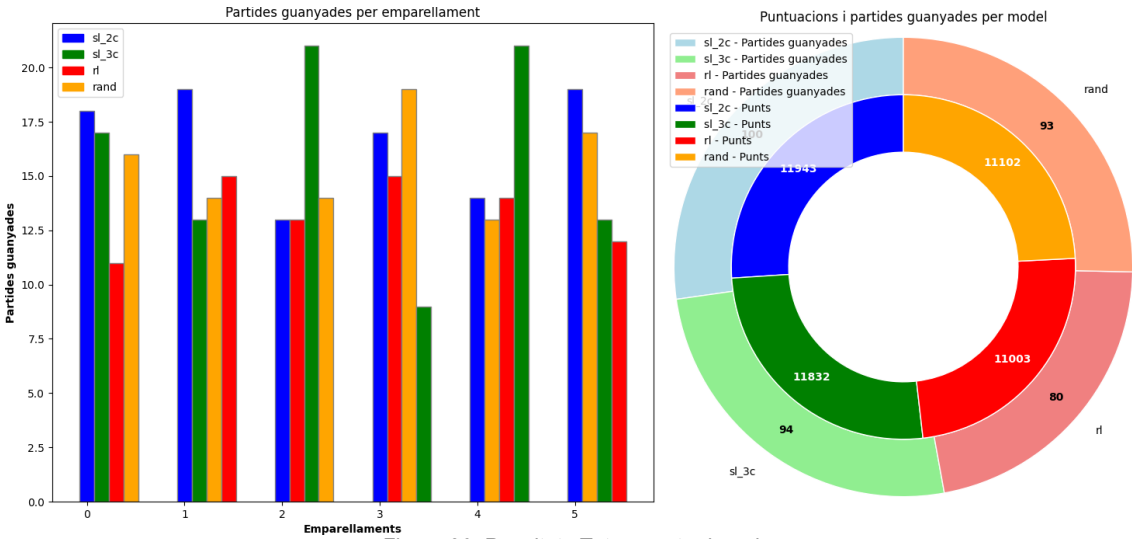

Figura 36: Resultats Tute - quatre jugadors

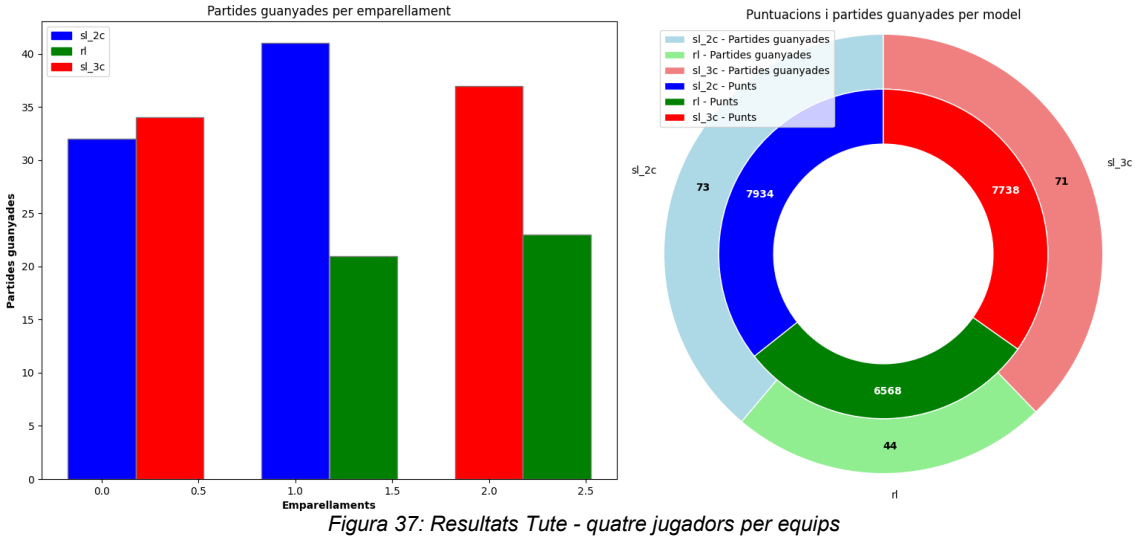

Figura 37: Resultats Tute - quatre jugadors per equips

7.4.3. Tute amb només obligació d'assistir

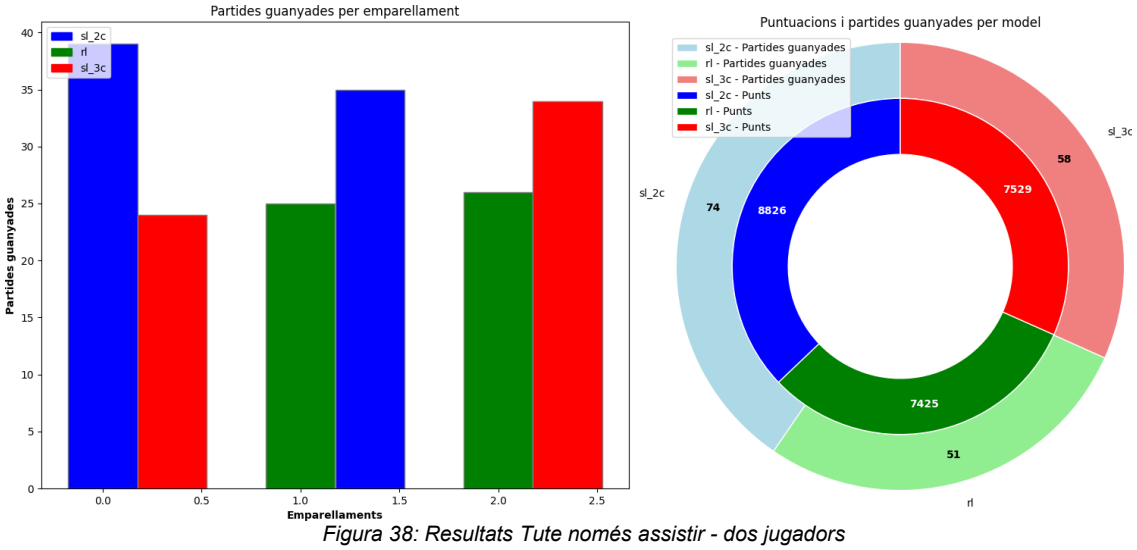

Figura 38: Resultats Tute només assistir - dos jugadors

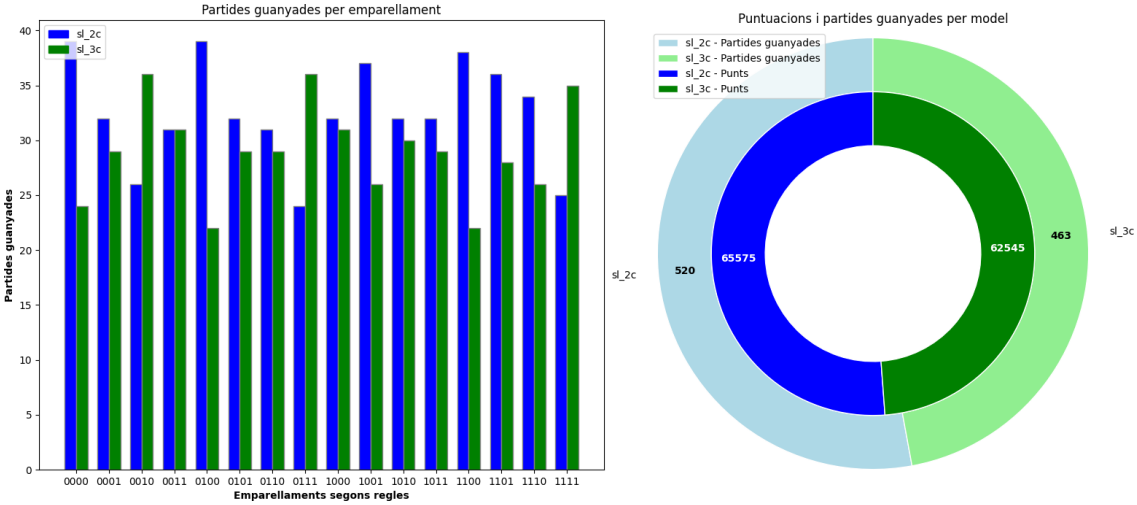

Figura 39: Resultats Tute només assistir - dos jugadors amb regles

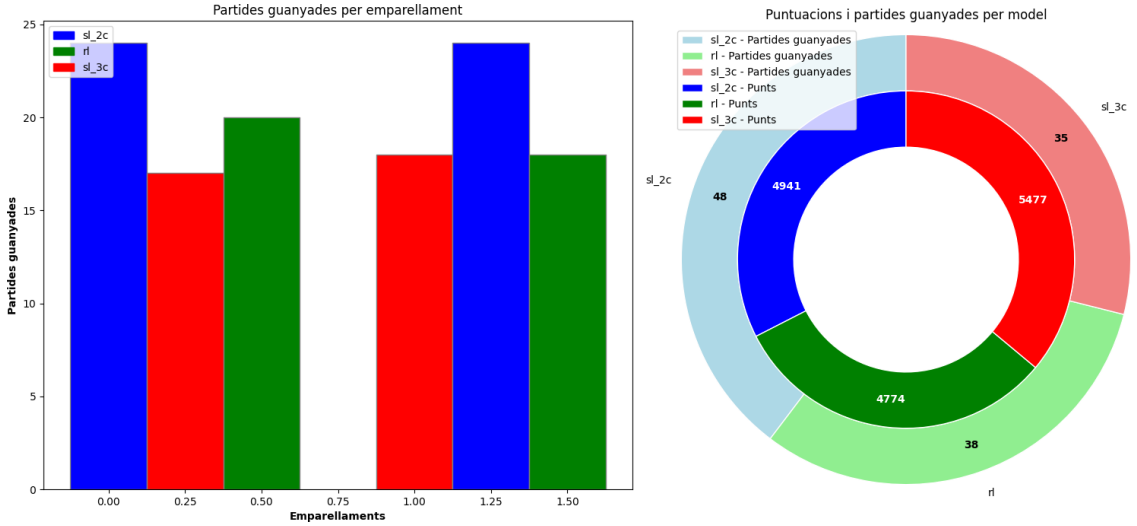

Figura 40: Resultats Tute només assistir - tres jugadors

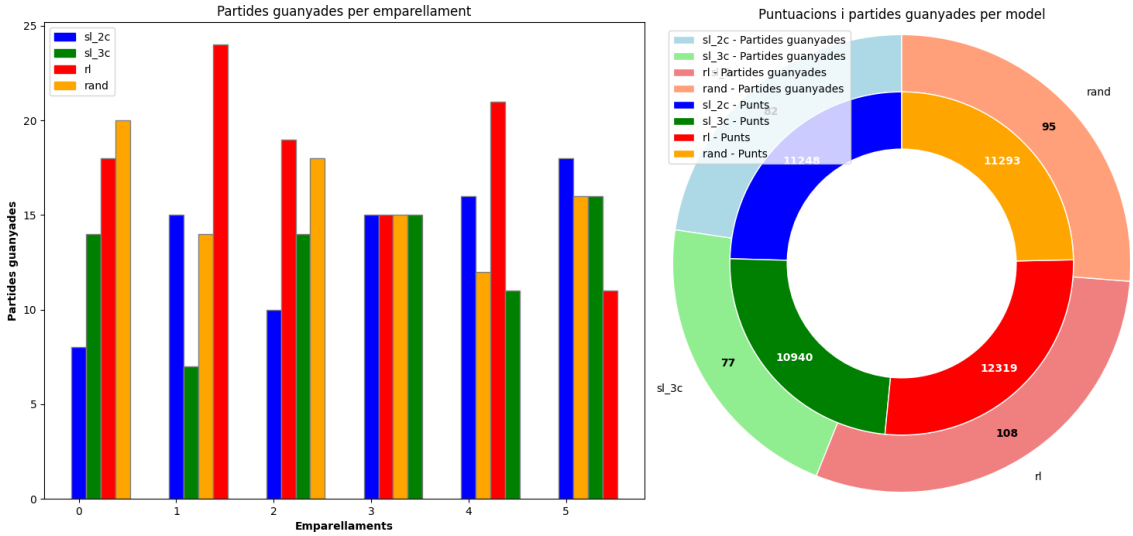

Figura 41: Resultats Tute només assistir - quatre jugadors

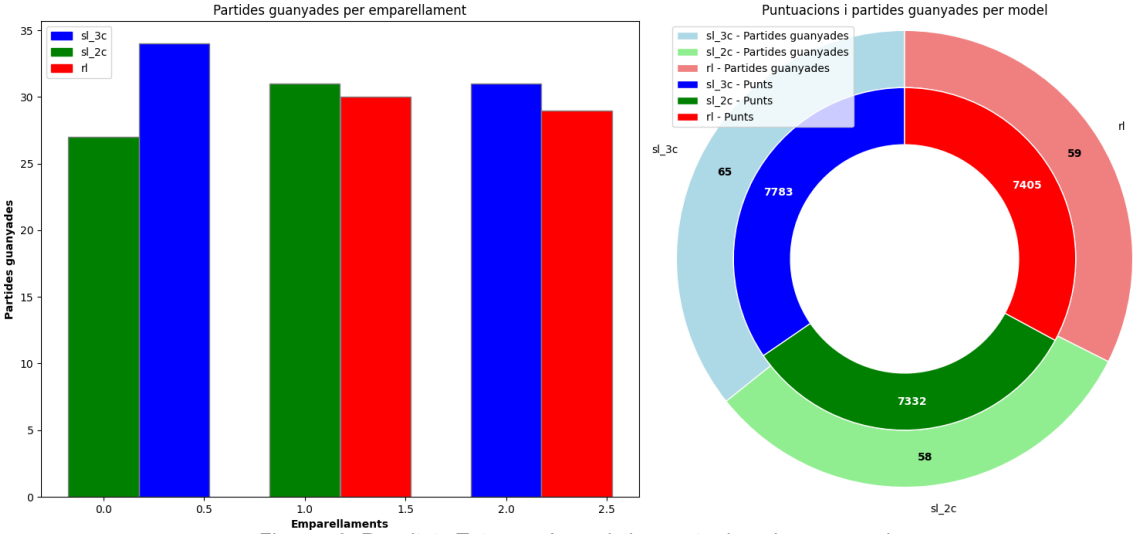

Figura 42: Resultats Tute només assistir - quatre jugadors per equips

8. Discussió

8.1. Anàlisi dels resultats

En aquest apartat es parlarà de “models”, en general, per referir-se a qualsevol dels models, individus oi agents que s'han obtingut, i es parlarà d'individu o agent per referir-se a algun d'ells de forma directa.

8.1.1. Brisca

Per a la modalitat de dos jugadors, el model supervisat de dues capes ha estat superior en tots els emparellaments, inclòs en els enfrontaments amb totes les combinacions de regles contra el de tres capes. No obstant això, aquesta superioritat s'ha revertit contra el model de tres capes a mesura que s'ha incrementat el nombre de jugadors.

Aquest canvi s'explica perquè l'arquitectura del model de tres capes no ha funcionat tan bé després de la normalització de les dades. Amb menys quantitat de dades d'entrada, la configuració de neurones per capa “inputs/2”, “inputs/4”, “inputs/6” resulta en capes amb poques neurones. En canvi, l'augment del nombre de jugadors també incrementa el nombre d'entrades, i, per tant, també ho fa el nombre de neurones de cada capa, millorant la seva capacitat de predicció.

És important tenir en compte que la funció heurística aplicada als models supervisats s'ha desenvolupat principalment per a la modalitat de dos jugadors. Això també pot explicar per què aquests models perden eficàcia en modalitats de més de dos jugadors i s'obtinguin resultats més igualats.

Per la seva banda, l'agent entrenat per reforç no ha mostrat un bon rendiment en cap dels emparellaments en la modalitat de dos i tres jugadors, excepte en l'emparellament contra l'individu genètic. No obstant això, en la modalitat de quatre jugadors, els resultats s'han igualat, ja que el joc es torna més aleatori, oferint avantatges als jugadors menys experts. Això és evident en les 71 victòries que ha aconseguit el jugador que pren decisions aleatòries.

A la modalitat per equips també s'observa aquesta igualtat, tot i que el component estratègic és superior per aquesta modalitat.

Durant les simulacions dels emparellaments, s'ha observat com l'agent entrenat per reforç ha escollit entre una i dues accions aleatòries per a la modalitat de dos jugadors (de 20 rondes), entre dues i cinc per a la de tres jugadors (de 13 rondes) i entre tres i set per a la de quatre jugadors (de 10 rondes).

El nombre d'estats augmenta amb l'increment del nombre dels jugadors, ja que hi ha més cartes jugades per ronda i, per tant, més estats possibles. En conseqüència, tot i entrenar els agents amb el mateix nombre d'episodis, això ha portat a una disminució en el percentatge d'estats visitats i a una major freqüència de decisions aleatòries, per culpa de la menor similitud entre les situacions del joc.

8.1.2. Tute

El joc del Tute té un estat del joc més ampli i complex que la Brisca, amb més cartes a les mans i la possibilitat per als jugadors de cantar. No obstant això, els jugadors estan limitats a jugar les cartes de la seva mà per l'obligació d'assistir, muntar, fallar i contrafallar. En moltes ocasions, només poden triar entre una i quatre cartes.

Les decisions més importants les pren el jugador que inicia la ronda. Això li permet seleccionar qualsevol carta de la seva mà, limitant les opcions dels seus rivals. Aquest factor és el més important a la modalitat de dos jugadors, mentre que l'augment en el nombre de participants complica la dinàmica del joc i els jugadors menys experimentats poden trobar-hi més facilitats.

Els dos models supervisats mostren un rendiment similar en les modalitats de dos i tres jugadors. Com que es disposa de més dades d'entrada que en la modalitat de la Brisca, el model de tres capes disposa de més neurones i rendeix millor.

En canvi, l'agent per reforç ha guanyat molt poques partides. En cada partida, realitza entre 12 i 15 accions aleatòries, de les 20 possibles, per a la modalitat de dos jugadors. La situació és similar a les modalitats de tres i quatre jugadors de la Brisca. L'agent ha visitat un percentatge menor, per tant, l'abstracció de l'estat també és menys efectiva.

En la modalitat de quatre jugadors, els models i l'agent mostren un rendiment similar. Tant el model de dues capes, com el de tres, així com el jugador que pren decisions aleatòries, han obtingut un nombre de victòries similars. L'agent per reforç també s'acosta a aquests resultats. Els bons resultats del jugador que pren decisions aleatòries demostren que l'experiència en el joc és menys determinant quan el nombre de jugadors augmenta, per aquest motiu els resultats estan més igualats en aquestes modalitats. En canvi, en la modalitat per equips, l'agent per reforç disminueix el seu rendiment.

En general, l'agent entrenat per reforç ha escollit entre 13 i 16 vegades una opció aleatòria en la modalitat de dos jugadors, de 8 a 10 en la de tres jugadors i de 3 a 7 en la de quatre jugadors. La majoria de les accions preses per al Tute han estat aleatòries, i això ha influenciat el seu rendiment general, sobretot en la modalitat de dos jugadors on l'experiència és més determinant.

8.1.2. Tute només assistir

El joc del Tute canvia totalment la seva dinàmica quan només s'aplica l'obligació d'assistir. Els jugadors tenen més llibertat per triar les cartes, la qual cosa implica un augment en la complexitat (on l'experiència és més determinant que en la versió original).

Curiosament, els resultats són més igualats en la versió de dos jugadors comparats amb la versió original. Tot i que el model de dues capes és lleugerament superior al de tres capes, l'agent entrenat per reforç ha igualat gairebé al de tres capes. A la versió original es podria esperar major igualtat perquè hi ha menys opcions per triar les cartes.

L'agent ha escollit el mateix nombre de jugades aleatòries que en el Tute, per tant, o bé els models supervisats no són tan eficaços en aquesta modalitat (potser la funció heurística no està ben adaptada) o bé l'agent per reforç ha tingut molta sort en el repartiment de les cartes.

En la modalitat de tres jugadors, s'observa un patró similar. Malgrat que l'agent entrenat per reforç ha aconseguit alguna victòria més que el de tres capes, el model de dues capes continua com el que més victòries ha aconseguit.

En la modalitat de quatre jugadors, es torna a percebre que la complexitat del joc fa disminuir l'experiència necessària per a l'obtenció de bons resultats. Destaquen l'agent entrenat per reforç i el jugador que pren decisions aleatòries com els que més victòries han aconseguit. Això es reprodueix també en la modalitat per equips.

El fet que els models supervisats no han demostrat un rendiment similar que a la Brisca o al Tute suggereix que cal revisar i ajustar la funció heurística per a tenir en compte les situacions d'aquesta modalitat que poden haver passat per alt.

8.2. Fortaleses i debilitats dels mètodes i enfocaments

8.2.1. Aprenentatge supervisat

Les xarxes neuronals són unes de les tècniques supervisades més utilitzades per a l'obtenció de models predictius, gràcies a la seva eficàcia i rapidesa en l'entrenament.

És factible dur a terme diverses proves i ajustar la configuració de paràmetres d'entrenament, com ara les funcions d'activació i d'optimització, les dades d'entrada, l'etiqueta de sortida, i l'arquitectura de la xarxa neuronal.

El rendiment dels models depèn en molta mesura de la qualitat de l'etiqueta de sortida. Quan l'etiqueta és correcta, els models mostren un rendiment elevat; en cas contrari, el rendiment pot ser baix. Aquest comportament s'ha pogut comprovar en les diverses variants utilitzades.

Per exemple, l'etiqueta de la variant “SL-WL” és massa ambigua, ja que assigna la mateixa sortida a totes les rondes d'un jugador en una partida, sense distingir entre jugades bones i dolentes. Això ha portat a models que no han après a jugar.

En canvi, l'etiqueta de la variant “SL-P” ha funcionat segons el previst, en la modalitat de dos jugadors. Utilitzant puntuacions positives i negatives, s'ha obtingut un model amb una estratègia clara per maximitzar la puntuació de cada ronda. No obstant això, quan només s'ha utilitzat la puntuació positiva, s'ha observat un comportament més agressiu en els torns on el model inicia la ronda.

Malauradament, aquesta variant no ha demostrat el mateix rendiment en modalitats amb més jugadors, pel fet que les puntuacions es desequilibren entre tots els jugadors, fent que l'etiqueta no sigui adequada per a altres modalitats.

Els models obtinguts amb la variant “SL-PH” han demostrat un bon rendiment i han demostrat que poden jugar bé en qualsevol de les modalitats i combinació de regles opcionals, gràcies a la diversitat de dades en el conjunt d'entrenament.

Els models ocupen poc espai físic, pel fet que només cal emmagatzemar els pesos i biaixos de les neurones. Això facilita la seva càrrega a memòria d'una manera molt ràpida.

Els models de dues capes han mostrat un bon rendiment després de la reducció de la dimensionalitat de les dades d'entrada. Aquesta optimització ha permès reduir el nombre de capes i neurones sense perdre eficiència, accelerant l'entrenament, pel fet que es disminueix la quantitat de càlculs durant les fases de propagació cap endavant i retropropagació.

En canvi, els models de tres capes només s'han mostrat competitius en les modalitats de més de dos jugadors i en el Tute, on el nombre d'entrades i el de neurones augmenta.

És important considerar que els models només poden imitar l'estratègia inclosa en el conjunt de dades durant l'entrenament, i no poden desenvolupar les seves pròpies estratègies. Això fa que siguin molt previsibles.

Per acabar, l'arquitectura seleccionada per les xarxes neuronals afecta el temps necessari per prendre decisions, pel fet que l'acció forma part de les dades d'entrada i no de la sortida. Per tant, és necessari realitzar tantes prediccions com accions possibles a cada torn.

8.2.2. Aprenentatge per reforç

El mètode Monte Carlo es caracteritza per ser un mètode que no necessita modelar l'entorn en el qual està interactuant. Els agents aprenen a partir de l'experiència acumulada al llarg dels episodis i actualitzen la seva política al final de cadascun d'ells. A diferència d'altres mètodes,

es té en compte totes les decisions preses des de l'inici fins al final de l'episodi, i no a cada decisió per separat.

S'han utilitzat dues recompenses que poden donar lloc a estratègies lleugerament diferents. Primer, s'ha considerat només la puntuació positiva per a cada ronda, on els agents intenten maximitzar la seva puntuació sense tenir en compte les jugades dolentes. Després, s'ha introduït la recompensa positiva i negativa, on els agents també poden tenir en compte les jugades menys encertades.

Inicialment, s'ha implementat la variant “MC-OS”, però no ha donat els resultats esperats. El problema és que no s'ha tingut en compte la quantitat d'estats diferents en el joc i la necessitat de visitar-los tots almenys una vegada.

Amb aquesta variant només s'ha aconseguit entrenar un agent amb 4.000.000 d'episodis, ja que els diccionaris creixen molt i consumeixen molta memòria RAM. No obstant això, aquest nombre d'episodis és insuficient per visitar tots els estats i que l'agent pugui aprendre a jugar en qualsevol situació.

S'ha intentat utilitzar un agent que jugués contra si mateix perquè sigui capaç d'acumular més informació en menys episodis, però, tot i ser efectiu, no ha estat suficient. També s'ha intentat reduir la informació que l'agent percep de l'entorn, i s'ha aconseguit reduir el nombre d'estats i la mida dels diccionaris. Però, aquesta reducció pot afectar negativament les estratègies que els agents poden desenvolupar.

Perquè l'agent sigui capaç de donar resposta a qualsevol situació, s'ha implementat la variant “MC-MS” i l'abstracció de l'estat del joc. Aquesta variant ha donat millors resultats, ja que permet als agents cercar accions en situacions similars. No obstant això, només ha funcionat per a la modalitat de dos jugadors de la Brisca, pel fet que el nombre d'estats augmenta considerablement quan augmenta la complexitat dels jocs.

La grandària dels diccionaris fa que tant l'emmagatzemament com la seva càrrega a memòria siguin molt lents. A mesura que els diccionaris creixien, es limita la quantitat d'episodis que es poden executar durant l'entrenament i es restringeix la quantitat d'informació que l'agent pot emmagatzemar dels diferents estats del joc. En cap cas s'ha aconseguit executar més de 3.000.000 d'episodis per aquesta variant.

S'ha intentat reduir la mida dels diccionaris fusionant els diccionaris Q i parelles visitades, ja que comparteixen les mateixes claus, però això no ha estat suficient. Si s'aconsegueix reduir dràsticament la mida dels diccionaris, es podran entrenar els agents amb molts més episodis.

8.2.3. Aprenentatge genètic

Aquesta tècnica està basada en l'optimització de les solucions mitjançant l'encreuament i la mutació dels individus de la població de forma variada i diversa. No obstant això, és necessari fer evolucionar una gran quantitat de generacions perquè els individus puguin convergir a solucions acceptables.

Tot i el seu potencial per generar individus amb estratègies molt diverses, no ha resultat adequada per a un problema tan complex com el que s'ha intentat resoldre. A més, com que no existeix una funció d'avaluació que permeti avaluar cada individu per separat d'una forma ràpida i eficient, s'han fet emparellaments entre els mateixos individus o utilitzant un altre model o agent com a rival per poder escollir els individus que més victòries i punts han aconseguit.

Aquesta forma d'avaluar implica que, a mesura que augmenta el nombre d'individus per població, s'han de simular més emparellaments i partides, limitant el nombre d'individus que es pot utilitzar per a l'entrenament.

D'altra banda, només és possible entrenar els individus per una combinació de regles opcionals concreta. S'ha de realitzar un entrenament per cadascuna de les combinacions per obtenir individus que puguin jugar amb qualsevol combinació de regles. No obstant això, es pot realitzar un entrenament inicial sense cap regla opcional, permetent que els individus aprenguin a jugar al joc bàsic. D'aquesta manera, es facilitaria l'entrenament per altres regles en un entrenament posterior utilitzant aquests individus com a població inicial.

Amb la variant "GA-XN", s'ha pogut replicar l'arquitectura de la xarxa neuronal utilitzada en l'entrenament supervisat. S'ha pogut prescindir d'utilitzar les accions com a dades d'entrada, utilitzant-les com a sortida de la xarxa neuronal, pel fet que no cal utilitzar cap etiqueta per a l'entrenament.

Aquest canvi ha permès que només calgui realitzar una predicció per cada torn en comptes de cada possible acció que l'individu pot realitzar. No obstant això, l'avaluació de la població ha resultat molt lenta a causa de temps elevat de les prediccions i la necessitat de simular tantes partides per emparellament.

Per accelerar el procés d'avaluació, s'han executat les simulacions dels emparellaments en paral·lel utilitzant diferents fils de la CPU. Però, tot i que s'ha aconseguit reduir significativament el temps total, no ha estat suficient per dur a terme entrenaments amb moltes generacions.

Amb la variant "GA-RL", s'ha substituït la representació de l'individu per un agent per reforç, ja que la seva predicció instantània durant el procés d'entrenament podria accelerar el procés d'avaluació. A més, l'encreuament i les mutacions dels diccionaris són una bona idea per maximitzar l'exploració, sobretot amb un alt nombre d'individus a la població.

Tot i aconseguir reduir el temps d'avaluació de cada generació, l'encreuament i la mutació dels diccionaris s'ha anat alentint amb el pas de les generacions, ja que els diccionaris creixen i requereixen cada vegada més temps per a la seva execució.

8.3. Limitacions

Cadascuna de les tècniques utilitzades ha presentat alguna limitació, afectant el rendiment final dels models o, fins i tot, obligant a abandonar l'entrenament amb la tècnica.

En l'aprenentatge supervisat, hi ha una limitació en el nombre màxim de partides amb el qual es pot entrenar els models. S'han pogut dur a terme proves d'entrenament de fins a 8.000 partides per combinació de regles (128.000 en total per a la modalitat de dos jugadors) sense cap problema. No obstant això, utilitzant un conjunt amb 10.000 partides per combinació (160.000 partides), no s'ha pogut finalitzar l'entrenament per falta de memòria RAM.

A més, els models només poden imitar el comportament inclòs en el conjunt de dades utilitzat i no poden aprendre les seves pròpies estratègies. Això els fa molt previsibles i els posa en desavantatge contra jugadors professionals, que poden aprofitar-ho en el seu benefici.

En l'aprenentatge per reforç, la mida dels diccionaris s'incrementa a mesura que l'agent interactua amb l'entorn i aprèn a jugar, i, a la vegada, s'incrementa l'ús de la memòria RAM, limitant el nombre d'episodis que es poden executar.

Els agents només poden ser entrenats per una combinació de regles opcionals concretes i no es pot reentrenar per una altra combinació.

L'aprenentatge genètic necessita una gran quantitat de generacions per obtenir solucions òptimes. No obstant això, el nombre de generacions que entrenades ha estat limitat per l'alt

temps en l'avaluació de la població amb la variant “GA-XN”, i per l'alt temps en l'execució de les funcions d'encreuament i mutació amb la variant “GA-RL”.

Tot i l'ús de l'execució dels emparellaments en paral·lel, existeix un límit en la seva eficàcia, ja que la CPU no pot executar simultàniament totes les accions.

9. Conclusions

9.1. Conclusions

L'entrenament utilitzant l'aprenentatge supervisat ha estat el més equilibrat. S'han aconseguit molt bons resultats amb un temps d'entrenament baix, i cada model es pot utilitzar per a qualsevol combinació de regles opcionals. A més, els models són molt lleugers i el temps de càrrega és molt ràpid. No obstant això, les prediccions són lentes i es requereix fer una predicció per cada possible acció que el jugador pot prendre en un moment donat de la partida.

De les tres variants implementades, només la variant “SL-PH” ha mostrat un bon rendiment en qualsevol modalitat. La funció heurística serveix de guia per a la presa de decisions dels models, però aquests només poden imitar l'estratègia definida per la funció i no poden desenvolupar les seves pròpies estratègies.

L'etiqueta utilitzada a la variant “SL-WL” és molt ambigua i no permet que els models aprenguin a diferenciar les jugades bones de les dolentes, fet que es veu accentuat per l'aleatorietat de les accions en el conjunt de dades.

D'altra banda, l'etiqueta utilitzada a la variant “SL-P” sí que ha funcionat bé per a les modalitats de dos jugadors, ja que la puntuació entre els jugadors està equilibrada. No obstant això, l'equilibri es perd quan s'afegeixen més jugadors, i els models es tornen erràtics i inconsistents en les prediccions.

Pel que fa a l'entrenament amb l'algoritme Monte Carlo, perquè els agents siguin capaços de predir qualsevol acció en qualsevol situació, han de visitar cada estat i acció, com a mínim una vegada. Això fa que en problemes molt complexos com el que s'ha volgut resoldre necessitin ser executats amb molts episodis.

Els agents poden aprendre les seves pròpies estratègies, però estan influenciades per la recompensa utilitzada.

Per a la variant “MC-OS” no s'ha tingut en compte que els jocs de la Brisca i el Tute tenen una quantitat enorme d'estats diferents, i que es requereix una gran quantitat d'episodis per aconseguir visitar-los tots.

A mesura que els agents aprenen, els diccionaris creixen molt, en bona part per culpa de la tria de les claus, que utilitza el valor decimal de la totalitat dels inputs binaris. Això ha provocat un alt ús de memòria RAM, limitant el nombre d'episodis que es poden executar i el total d'estats visitats.

La variant “MC-MS” ha intentat resoldre aquest problema creant una abstracció de l'estat. Tot i que aquesta solució ha funcionat per a les modalitats de dos jugadors, no ha estat suficient per a la resta de modalitats. L'abstracció permet donar resposta a un ampli nombre de situacions que no s'han visitat explícitament. No obstant això, l'augment del nombre de jugadors fa augmentar el total d'estats i redueix el nombre de situacions similars que es poden trobar amb l'abstracció.

A més, aquest procés de cerca fa que la tria d'una acció sigui molt lenta, i això impedeix que els agents puguin ser utilitzats en partides reals.

D'altra banda, l'entrenament utilitzant l'aprenentatge genètic és la tècnica que més llibertat dona als individus per aprendre les seves pròpies estratègies i té molt potencial. No obstant això, aquesta tècnica no està pensada per a problemes tan complexos com el que s'ha volgut resoldre.

El fet que no existeixi una forma d'avaluar la població d'una manera simple ha estat un punt crític per a la variant “GA-XN” i ha alentit tot el procés d'entrenament. Les prediccions dels individus són lentes i s'ha hagut de simular múltiples partides amb cada individu per aconseguir una avaluació imparcial.

Tot i que l'execució d'emparellaments en paral·lel ha reduït molt el temps d'avaluació, s'ha trobat que hi ha un límit. Quan s'augmenta molt el nombre de fils, la reducció de temps desapareix.

En resum, és molt probable que els enfocaments aplicats a cadascuna de les tècniques no siguin els millors, i hi ha molt marge de millores per a futures investigacions.

9.2. Línies de futur

Per a les tècniques d'aprenentatge supervisat, ha quedat pendent dur a terme experiments amb diferents funcions d'optimització i d'activació. Seria important comprovar si aquests canvis milloren els resultats obtinguts per la variant “SL-PH”.

A més, cal investigar si és possible utilitzar altres etiquetes de sortida, o si es pot fer alguna combinació amb les etiquetes de les variants “SL-WL” i “SL-P” per obtenir alguna altra variant que generi una estratègia diferent.

També es podrien implementar més funcions heurístiques, cadascuna seguint una estratègia diferent, i generar un nou conjunt de dades utilitzant informació de partides que incloguin totes les funcions. Això permetria realitzar de nou els entrenaments per comprovar si els models són capaços d'utilitzar una estratègia o una altra segons el desenvolupament de la partida.

Finalment, es podria recopilar un conjunt de dades a partir de jugades de jugadors professionals i experimentats i realitzar un entrenament per a cadascuna de les variants. Seria interessant comprovar si el rendiment millora per la variant “SL-PH” o si els models pateixen de sobreajustament, de la mateixa manera que quan s'ha generat un conjunt de dades utilitzant les accions del model supervisat.

Quant a les tècniques d'aprenentatge per reforç, l'objectiu principal hauria de ser l'optimització en l'ús de la memòria RAM per poder entrenar un agent en qualsevol de les seves variants, utilitzant molts més episodis, de manera que pugui recopilar més informació.

Aquesta fita es podria aconseguir reduint el pes dels diccionaris, ja que la major part d'aquest pes és degut al valor decimal de l'estat associat a la clau. Per tant, es podria intentar reduir la representació de la clau codificant el seu valor, per exemple, utilitzant una codificació hexadecimal o superior.

A més, es podria utilitzar un enfocament diferent, com l'ús de l'abstracció de l'estat com a clau dels diccionaris. D'aquesta manera, es reduiria el nombre total d'estats i la mida dels diccionaris considerablement, eliminant la necessitat fer la cerca de situacions similars, ja que el diccionari únicament contindria aquesta abstracció. Així, els diccionaris serien menys pesats i requeririen menys recursos, donant pas a la possibilitat d'augmentar el nombre d'episodis per entrenament. No obstant això, es perdria varietat en les estratègies que poden aprendre, pel fet que l'agent disposaria de menys informació sobre el joc.

La investigació ha deixat pendent la realització d'experiments amb altres valors d'exploració i de decreixement de l'exploració, així com la utilització d'altres recompenses. Per exemple, es podria utilitzar com a recompensa la suma de la puntuació i el valor de la funció heurística.

Per acabar, s'hauria d'optimitzar la cerca de situacions similars utilitzant l'abstracció de l'estat del joc, ja que actualment és una operació molt lenta. Inicialment, es considerava fer aquesta cerca progressivament, cercant situacions de més similars a menys, però, finalment s'ha deixat la màxima abstracció possible per reduir el temps de predicció.

Per a les tècniques d'aprenentatge genètic, caldria investigar altres enfocaments que permetin avaluar la població de manera més ràpida que a la variant “GA-XN”, però que, alhora, permetin executar les tasques d'encreuament i mutació més ràpidament que en la variant “GA-RL”.

A més, la investigació ha deixat pendents experiments que permetin avaluar el rendiment de les funcions d'encreuament i mutació, així com el de les directives de control i la variació del nombre d'individus per població.

Un cop s'hagi aconseguit executar un entrenament i obtenir individus amb un bon rendiment, caldria comprovar si és factible utilitzar aquests individus com a població inicial d'un altre entrenament amb unes altres regles opcionals.

9.3. Seguiment de la planificació

La planificació del treball s'ha pogut seguir sense contratemps durant les fases inicials, i, fins i tot, s'ha aconseguit avançar en alguna de les fases. Per exemple, es va finalitzar la implementació de l'entorn del joc i de la GUI unes setmanes abans del previst, i això ha permès dedicar més temps a altres fases més importants, com són la implementació dels algoritmes i l'entrenament dels agents.

Per altra banda, s'ha requerit més temps del previst per a l'entrenament dels models, ja que han aparegut limitacions i problemes que han obligat a modificar la implementació dels algoritmes i els enfocaments utilitzats.

A l'anàlisi de riscos ja es va preveure que l'entrenament podria ser més lent del que es pensava, i que algunes tècniques i entrenaments finals no podrien finalitzar. Per exemple, l'entrenament amb algoritmes genètics s'ha abandonat perquè s'han necessitat cinc dies per entrenar 1.000 generacions en només una de les modalitats, sense obtenir bons resultats.

Tampoc s'han pogut finalitzar tots els entrenaments previstos per a l'entrenament per reforç, ja que només s'ha pogut obtenir agents que sense aplicar cap regla opcional. L'alt temps necessari per a l'entrenament ha impossibilitat finalitzar aquests entrenaments.

10. Glossari

10.1. Glossari tecnològic

Agent – En intel·ligència artificial, és un sistema computacional dissenyat per a interactuar amb el seu entorn amb el fi d'assolir certs objectius o resoldre problemes específics.

AI – Acrònim de "**Artificial Intelligence**".

Algoritme genètic – Tècnica d'optimització de l'aprenentatge computacional inspirada en l'evolució natural que utilitza conceptes com la selecció natural, encreuament i mutació per a trobar solucions òptimes o properes a problemes complexos.

Aprenentatge computacional – Camp de la intel·ligència artificial que se centra en el desenvolupament d'algoritmes i tècniques que permeten a les computadores aprendre a partir de dades, identificar patrons o prendre decisions sense ser programades explícitament per a la tasca específica.

Aprenentatge per reforç – Tècnica de l'aprenentatge computacional on un agent interactua amb l'entorn amb el fi de maximitzar una recompensa acumulativa en forma de retroalimentació rebuda d'acord amb les seves accions al llarg de la interacció.

Aprenentatge profund – Camp de la intel·ligència artificial que fa ús de xarxes neuronals profundes per aprendre i extreure representacions complexes de dades.

Artificial intelligence – Veure "**Intel·ligència artificial**".

Big data – Veure "**Grans conjunts de dades**".

Deep learning – Veure "**Aprenentatge profund**".

Deep neural network – Veure "**Xarxa neuronal profunda**".

DL – Acrònim de "**Deep Learning**".

Entorn del joc – Espai simulat on es desenvolupen les partides dels jocs.

Interfície gràfica d'usuari – Entorn on els usuaris poden interactuar amb una computadora per mitjà d'elements visuals.

Generació – En algoritmes genètics, és una població completa de possibles solucions per un problema donat.

Graphical user interface – Veure "**Interfície gràfica d'usuari**".

Grans conjunts de dades – Col·leccions massives d'informació que superen la capacitat de processament de les eines tradicionals.

GUI - Acrònim de "**Graphical User Interface**".

Intel·ligència artificial – Camp de la informàtica que s'enfoca en el desenvolupament de sistemes capaços d'executar tasques que normalment requereixen intel·ligència humana.

Machine Learning – Veure "**Aprenentatge computacional**".

MiniMax – Tècnica de la intel·ligència artificial per a la presa de decisions en jocs de dos jugadors que es basa en la cerca exhaustiva de totes les possibles jugades i les seves conseqüències, amb l'objectiu de maximitzar el benefici propi mentre es minimitza el del rival.

ML – Acrònim de "**Machine Learning**".

Neural network – Veure "Xarxa neuronal"

Neurona – En xarxes neuronals, és la unitat bàsica de processament que simula el funcionament d'una neurona biològica.

NN – Acrònim de "**Neural Network**".

Perceptró – Xarxa neuronal d'una sola capa que pot aprendre a classificar dades linealment separables.

Població – En algoritmes genètics, és una possible solució per un problema donat.

Poda alfa-beta – Tècnica d'optimització utilitzada en arbres de cerca que permet reduir el nombre de nodes a avaluar.

Recompensa – En aprenentatge per reforç, és la retroalimentació, positiva o negativa, que rep l'agent després de cada acció.

Reinforcement learning – Veure "**Aprenentatge computacional**"

RL – Acrònim de "**Reinforcement Learning**"

Xarxa neuronal – Model computacional inspirat en el cervell humà, que està compost per capes de neurones interconnectades entre si.

Xarxa neuronal profunda – Xarxa neuronal composta per moltes capes intermèdies de neurones.

10.2. Glossari de la Brisca i el Tute

Assistir – Al tute, és la denominació de l'acció d'un jugador que disposa a la seva mà d'alguna carta del **coll de ronda**.

Cantar – Al tute, quan un jugador disposa del Rei i el Cavall d'un mateix **coll** i informa a la resta de jugadors.

Coll – Una de les quatre categoríes en les que es divideix la baralla Espanyola (ors, bastos, espases i copes).

Coll del triomf – **Coll** amb major preferència que la resta durant una partida.

Coll de ronda – **Coll** de la primera carta en una **ronda**.

Contrafallar – Al tute, és la denominació de l'acció d'un jugador que no pot **assistir** ni **fallar.**

Fallar – Al tute, és la denominació de l'acció d'un jugador que no pot **assistir** i disposa a la seva mà d'alguna carta del **coll del triomf**.

Intercanviar – Acció d'intercanviar la carta de **triomf** per una carta de la mà.

Les 20 en "coll" – Al tute, acció de **cantar** de qualsevol **coll** diferent del **coll del triomf**.

Les 40 – Al tute, acció de **cantar** del **coll del triomf**.

Les 10 d'últimes – Regla que otorga 10 punts addicionals al guanyador de l'última ronda de la partida.

Mà negra – Quan un jugador disposa de l'As, el 3 i el Rei del **coll del triomf** a la seva mà.

Muntar – Al tute, és la denominació de l'acció d'un jugador que disposa a la seva mà d'alguna carta del **coll de ronda** amb major preferència que la carta amb major preferència de la ronda.

Repartidor – Jugador que reparteix les cartes a l'inici d'una partida.

Restricció – Al tute, normes que els jugadors han de seguir per jugar una carta (**assistir**, **muntar**, **fallar, trepitjar** i **contrafallar**)

Robar – Acció d'agafar una carta encara no jugada de la baralla.

Ronda – Part del joc on cada jugador ha de jugar una carta.

Valor – Puntuació associada a una carta.

Trepitjar – Al tute, és la denominació de l'acció d'un jugador que no disposa a la seva mà de cap carta del **coll de ronda**, però si disposa de cartes del **coll del triomf** amb major preferència que la carta d'un altre altre jugador que també ha **fallat** o **trepitjar**.

Triomf – Carta escollida a l'inici de la partida que indica el **coll** amb major preferència de la partida

Tute de cavalls o reis – Al tute, quan un jugador disposa dels quatre Cavalls o Reis a la seva mà.

11. Bibliografia

[1] **Jet New - Medium**. *A Summary of Alan Turing's Computing Machinery and Intelligence*. https://medium.com/@jetnew/a-summary-of-alan-m-turings-computing-machinery-and-intelligence-fd714d187c0b (02/03/2024)

[2] **Ian Sample - The Guardian**. *Race to AI: the origins of artificial intelligence, from Turin to ChatGPT*. https://www.theguardian.com/technology/2023/oct/28/artificial-intelligence-origins-turing-to-chatgpt (02/03/2024)

[3] **James V. Stone - Medium**. *A very Short History of Artificial Neural Networks.* https://jimstone-68634.medium.com/a-very-short-history-of-artificial-neural-networks-9820dfd6d903 (02/03/2024)

[4] **Miguel Sotaquirá - Codificando bits**. *Historia y evolución del aprendizaje por refuerzo*. https://www.codificandobits.com/curso/aprendizaje-por-refuerzo-nivel-basico/3-historia-aprendizaje-por-refuerzo/ (02/03/2024)

[5] **Sourabh Katoch, Sumit Singh Chauhan i Vijay Kumar – National Center for Biotechnology Information**. *A review on genetic algorithm: past, present and future*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7599983/ (02/03/2024)

[6] **Chris Piech - Standford**. *Deep Blue*. https://stanford.edu/~cpiech/cs221/apps/deepBlue.html (02/03/2024)

[7] **Amphy**. *The Rise of Chess AI: From Deep Blue to AlphaZero*. https://blog.amphy.com/deep-blue-alphazero/ (02/03/2024)

[8] **Chess Programming Wiki**. *AlphaZero*. https://www.chessprogramming.org/AlphaZero#:~:text=The%20algorithm%20is%20a%20more,used%20in%20classical%20chess%20programs. (02/03/2024)

[9] **Jose García - Xataka**. *El único jugador de Go que ha conseguido derrotar una vez a la IA de Google se retira porque "no puede ser derrotada"*. https://www.xataka.com/inteligencia-artificial/unico-jugador-go-que-ha-conseguido-derrotar-vez-a-ia-google-se-retira-porque-no-puede-ser-derrotada (02/03/2024)

[10] **DeepMind**. *AlphaGo*. https://deepmind.google/technologies/alphago/ (02/03/2024)

[11] **The AlphaStar Team - DeepMind**. *AlphaStar: Mastering the real-time strategy game StarCraft II*. https://deepmind.google/discover/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/ (02/03/2024)

[12] **Wikipedia**. *Brisca*. https://es.wikipedia.org/wiki/Brisca (09/03/2024)

[13] **Wikipedia**. *Tute*. https://es.wikipedia.org/wiki/Tute (09/03/2024)

[14] **Noam Brown and Tuomas Sandholm - Science .** *Superhuman AI for heads-up no-limit poker: Libratus beats top professionals.* https://www.science.org/doi/10.1126/science.aao1733 (12/04/2024)

[15] **Noam Brown and Tuomas Sandholm - Science .** Superhuman AI for multiplayer poker. https://www.science.org/doi/10.1126/science.aay2400 (12/04/2024)

[16] **Ankit Choudhary - Analytics Vidhya.** *Reinforcement Learning: Monte Carlo in Reinforcement Learning*. https://www.analyticsvidhya.com/blog/2018/11/reinforcement-learning-introduction-monte-carlo-learning-openai-gym/ (28/04/2024)

[17] **Bechir Trabelsi - Medium.** *On-policy vs Off-policy Monte Carlo Control Methods for Supply Chain Optimization: A Use Case of Inventory Replenishment*. https://bechirtr97.medium.com/on-policy-vs-off-policy-monte-carlo-control-methods-for-supply-chain-optimization-a-use-case-of-7b8d0d2e80e6 (28/04/2024)

[18] **Narcís P.R. - *Github***. *PyBrisca*. https://github.com/narcispr/pyBrisca

[19] **Bryan Collazo - Github.** *Briscas*. https://github.com/bcollazo/briscas

[20] **TensorFlow.** https://www.tensorflow.org/

[21] **Tkinter.** https://docs.python.org/3/library/tkinter.html

[22] **Owen Elliot - LinkedIn.** *Training a Neural Network with a Genetic Algorithm*. https://www.linkedin.com/pulse/training-neural-network-genetic-algorithm-owen-elliott (16/04/2024)

[23] **Ali Karazmoodeh – LinkedIn.** *Crossovers in genetic algorithms*. https://www.linkedin.com/pulse/crossovers-genetic-algorithms-ali-karazmoodeh-tthjf (16/04/2024)

[24] **Ali Karazmoodeh – LinkedIn.** Mutations *in genetic algorithms*. https://www.linkedin.com/pulse/mutations-genetic-algorithms-ali-karazmoodeh-u94pf (16/04/2024)

[25] **Ahmad Hassanat – Multidisciplinary Digital Publishing Institute.** *Choosing Mutation and Crossover Ratios for Genetic Algorithms - A Review with a New Dynamic Approach*. https://www.mdpi.com/2078-2489/10/12/390 (16/04/2024)

[26] **A.J. Umbarkar – ICTACT Journal on soft computing**. *Crossover operators in Genetic Algorithms: A Review*. https://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf (16/04/2024)

[27] **Tymothy Revell – NewScientist.** *AI just won a poker tournament againstm professional players*. https://www.newscientist.com/article/2119815-ai-just-won-a-poker-tournament-against-professional-players/ (12/05/2024)

[28] **Carnegie Mellon University.** *Carnegie Mellon and Facebook AI Beats Professionals in Six-Player Poker*. https://www.cmu.edu/news/stories/archives/2019/july/cmu-facebook-ai-beats-poker-pros.html (12/05/2024)

12. Annexos

12.1. Accés al repositori GitHub

L'accés al repositori GitHub és públic i es pot accedir mitjançant la següent URL: https://github.com/rarjonilla/rarjonilla_TFG

El repositori inclou tot el codi necessari per a l'entrenament utilitzant les diferents tècniques implementades, i també inclou els models finals per a l'entrenament supervisat i genètic (els de l'aprenentatge per reforç tenen una mida massa gran i no s'han pogut afegir al repositori).

12.2. Preparació de l'entorn de desenvolupament

Tota la investigació s'ha desenvolupat des d'un equip amb Windows 10 i amb l'IDE PyCharm.

El primer pas és descarregar i instal·lar l'última versió de PyCharm des del seu lloc web (en aquest cas s'ha descarregat la versió gratuïta): https://www.jetbrains.com/pycharm/

El segon pas és instal·lar les eines necessàries per a la configuració de l'entorn.

- Microsoft Visual C++ Redistributable: https://support.microsoft.com/help/2977003/the-latest-supported-visual-c-downloads
- Miniconda: https://docs.conda.io/en/latest/miniconda.html

El tercer pas és crear i activar l'entorn.

- Creació i activació de l'entorn: S'ha d'obrir l'aplicació Miniconda (és una línia de comandes) i executar les següents instruccions.
  - conda create --name **nom_entorn** python=3.9
  - conda activate **nom_entorn**
- Des de Pycharm, s'ha d'afegir el nou intèrpret (l'entorn creat prèviament).
  - Obrir PyCharm i clicar a l'entorn actual situat a la part inferior dreta de la finestra.
  - Seleccionar “Add interpreter -> Add local interpreter”.
  - Buscar i seleccionar l'entorn de miniconda creat prèviament.

El quart pas és instal·lar totes les llibreries necessàries.

- Activació de l'ús de la GPU (opcional): conda install -c conda-forge cudatoolkit=11.2 cudnn=8.1.9
- Actualitzar PIP: pip install --upgrade pip
- TensorFlow (les versions posteriors a 2.10 no permeten utilitzar la GPU per Windows): pip install "tensorflow<2.11"
- Pillow: conda install pillow
- SciKit: pip install -U scikit-learn
- Pandas: pip install pandas
- MyPy: pip install mypy
- Swig: conda install swig
- MatPlotLib: pip install matplotlib
- Tqdm: pip install tqdm

Si s'han realitzat la instal·lació opcional per activar la GPU, cal executar la següent instrucció per comprovar si s'ha activat correctament:

```
python -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

Si l'execució retorna una llista buida, hi ha alguna cosa que no ha anat bé. S'ha de comprovar que els drivers instal·lats de la targeta gràfica estan en la seva última versió.

12.3. Ús del programa

Per a l'execució de l'eina desenvolupada només cal executar el fitxer “main.py” i seguir les instruccions de la consola.

1. Simulació de partides: permet simular partides des del backend utilitzant qualsevol dels agents entrenats prèviament o utilitzant un agent aleatori.
2. Simulació de partides GUI: permet simular partides des de la GUI implementada utilitzant qualsevol dels agents entrenats prèviament o utilitzant un agent aleatori. Es pot escollir que el jugador 0 sigui un humà o no.
3. Entrenament d'un agent: permet entrenar un agent utilitzant qualsevol de les tècniques desenvolupades
    1. Agent supervisat amb dades normalitzades utilitzant l'etiqueta de sortida “Win or Lose”.
    2. Agent supervisat amb dades normalitzades utilitzant l'etiqueta de sortida “Puntuació”.
    3. Agent supervisat amb dades normalitzades utilitzant l'etiqueta de sortida “Puntuació + valor heurístic”.
    4. Agent genètic utilitzant l'aproximació de les xarxes neuronals.
    5. (pendent) Agent genètic utilitzant l'aproximació dels agents per reforç. Es pot executar manualment.
    6. (pendent) Agent per reforç utilitzant un sol estat. Es pot executar manualment.
    7. (pendent) Agent per reforç utilitzant múltiples estats. Es pot executar manualment.
4. (pendent) Simulacions: Permet realitzar simulacions i emmagatzemar en un fitxer CSV els resultats obtinguts. Es pot executar manualment amb el fitxer “/results/generate_csv.py”.
5. (pendent) Resultats: Permet generar les gràfiques a partir dels fitxers CSV generats del punt anterior. Es pot executar manualment amb el fitxer “/results/generate_graphs.py”.
6. Sortir: Finalitza l'execució del programa.

12.4. Linies de codi per a l'execució manual dels entrenaments.

12.4.1. Entrenament genètic (variant “GA-XN”)

```
# Les regles aplicades, posar a True per activar-la
rules = {'can_change': False, 'last_tens': False, 'black_hand': False, 'hunt_the_three': False, 'only_assist': False}
# total de jocs
total_games = 1
# Nombre de generacions
generations = 1000
# poblacio
population = 24
# millor població
best_population = 2
# directiva
directive = 3
# fils
threads = 6
# capes llista amb les neurones per capa
layers = [75, 50]
# doble seleció
double_selection = False
# selecció d'elit
elite_selection = False
# 1.Brisca, 2.Tute
game_type = 1
# 2, 3 o 4 jugadors
num_players = 2
```

```
# True. Individual, False. Equips (només per 4 jugadors)
single_mode = True
# Nom de la carpeta on es guardara l'entrenament
directory_name = 'nom_carpeta_on_es_guarda'


Genetic_training(game_type, total_games, num_players, single_mode, rules, False, False, directory_name, layers, population,
best_population, generations, 0, directive, None, None, True, threads, double_selection, None, elite_selection)
```

12.4.2. Entrenament genètic (variant “GA-RL”)

```
# Les regles aplicades, posar a True per activar-la
rules = {'can_change': False, 'last_tens': False, 'black_hand': False, 'hunt_the_three': False, 'only_assist': False}
# Nombre de generacions
generations = 1000
# poblacio
population = 12
# millor població
best_population = 4
# directiva
directive = 3
# fils
threads = 6
# doble selecció
double_selection = False
# selecció d'elit
elite_selection = False
# 1.Brisca, 2.Tute
game_type = 1
# 2, 3 o 4 jugadors
num_players = 2
# True. Individual, False. Equips (només per 4 jugadors)
single_mode = True
# Nom de la carpeta on es guardara l'entrenament
directory_name = 'nom_carpeta_on_es_guarda'
# Valor eps
eps = 0.1
# Valor decreixement eps
dec_eps = 1e-7
# valor de descompte
gamma = 1


Genetic_training_rl(game_type, total_games, num_players, single_mode, rules, False, False, directory_name, population,
best_population, generations, 0, directive, None, None, True, threads, double_selection, None, elite_selection, eps, dec_eps,
gamma, True)
```

12.4.3. Entrenament per reforç

```
# Les regles aplicades, posar a True per activar-la
rules = {'can_change': False, 'last_tens': False, 'black_hand': False, 'hunt_the_three': False, 'only_assist': False}
# Nombre d'episodis
episodes = 2000000
# 1.Brisca, 2.Tute
game_type = 1
# 2, 3 o 4 jugadors
num_players = 2
# True. Individual, False. Equips (només per 4 jugadors)
single_mode = True
# Ruta on es vol emmagatzemar l'agent
path = f'rl_models/tute_only_assist/2j/nom_carpeta_on_es_guarda'
# Valor eps
eps = 0.1
# Valor decreixement eps
dec_eps = 1e-7
# valor de descompte
gamma = 1
# True. Només 1 agent. False. Múltiples agents
# En cas de false, cal especificar el path de cadascun dels jugadors i substituir el None)
only_one_agent = True
# True. Multiples estats. False. Un estat
multiple_state = True
```

```
Reinforcement_training(game_type, episodes, num_players, single_mode, rules, [path, None, None, None], eps, dec_eps, gamma,
only_one_agent, multiple_state)
```