|  |
| --- |
| CIC Batxillerats |
| [Título del documento] |
| Treball de Recerca |

|  |
| --- |
| Marc Perez Fusco, Jan Ferrer Paramio, Teo Clerici Jurado  26-9-2024 |

**ÍNDEX**

[ABSTRACT 2](#_Toc178197157)

[INTRODUCCIÓ 2](#_Toc178197158)

[COM FUNCIONA LA APFLY-IA 2](#_Toc178197159)

[LA APFLY-IA DINS DEL CONTEXT “*FLAPPY BIRD”* 4](#_Toc178197160)

[FLAPPY BIRD 4](#_Toc178197161)

[FITNESS 4](#_Toc178197162)

[INPUTS 4](#_Toc178197163)

[ALTRES CONSTANTS 5](#_Toc178197164)

[L’APRENENTATGE LOGARITMIC 5](#_Toc178197165)

[COM AFECTEN LES VARIABLES 6](#_Toc178197166)

[CONCLUSIÓ 10](#_Toc178197167)

[LÍNIES A FUTUR 10](#_Toc178197168)

[BIBLIOGRAFIA 12](#_Toc178197169)

# ABSTRACT

En aquest treball, explorem les possibilitats de la Intel·ligència Artificial (IA) per a jugar al joc “*Flappy Bird*” utilitzant algoritmes avançats d'aprenentatge adaptatiu. L'ús combinat de mutacions i altres factors permet l'obtenció d'un model neuronal més eficient i capaç de realitzar càlculs ràpids. El model neuronal desenvolupat en aquest treball és capaç d’aprendre i adaptar-se a les condicions del joc. En aquesta recerca es demostra com la corba de l’aprenentatge és logarítmica i depèn fortament dels diferents comportaments de la IA a partir de les variables de mutacions, inputs i població.

# INTRODUCCIÓ

Darrerament, les tècniques neuroevolutives s’han popularitzat molt en el món de la IA, i en aqueta recerca hem volgut aplicar aquestes tècniques a un videojoc anomenat “*Flappy Bird”,* molt conegut entre els aficionats als videojocs. El treball vol mostrar com aprèn una IA que juga a un joc poc complex i com es pot millorar aquest aprenentatge, a partir de la variació de les condicions inicials del joc, per això s’analitza com de logarítmic és la corba de generació-fitness segons la configuració establerta variant configurant la població, els inputs i els tipus de mutació.

A partir de l’anàlisi de les dades obtingudes d’aquest aprenentatge, també es planteja trobar aquells paràmetres que fan que la IA realitzi la tasca de la millor manera, així com aquelles condicions més eficaces per entrenar-la, com ara quants agents són òptims per el seu aprenentatge i quantes generacions són necessàries per entrenar una IA per a que pugui dur a terme la tasca encomanada de forma perfecta.

Per obtenir els resultats que ens han permès dur a terme a aquest anàlisi, s’ha construït a mode de laboratori digital un “Flappy Bird” específic en Godot per poder inserir-hi una IA que hem creat exclusivament per aquest propòsit. Aquesta IA l’anomenem Autonomous Player Fast Learning Yeld (APFLY-IA)

# COM FUNCIONA LA APFLY-IA

[[1]](#footnote-2)Les xarxes neuronals són un sistema de ML (Machine Learning o Aprenentatge Automàtic) que és basa en la biologia, en l’estructura neuronal dels animals. Aquestes xarxes és representen en forma de graf, on els vèrtexs són les neurones encarregades d’emmagatzemar la informació i on les arestes són les connexions (amb una relació d’1:1) encarregades de transformar aquesta informació i transmetre-la a les neurones connectades. La primera columna, vermella amb títol d’*Input* són les neurones on col·locarem la informació necessària per fer funcionar la xarxa neuronal; la segona columna, blava amb títol *Hidden,* són les neurones encarregades d’emmagatzemar la informació que encara no s’ha acabat de processar (per problemes que són lineals aquesta capa és prescindible), i la tercera columna, verda amb títol Output, són les neurones encarregades de proporcionar les dades de sortida. Per saber el valor d’una neurona s’ha de sumar les multiplicacions entre les neurones estrictament precedents i la seva connexió.

En cas d’un problema no lineal els resultats de les neurones ocultes s’haurà de passar per una funció d’activació, com pot ser “sigmoide” i aquest ha sigut el cas del APFLY-IA, que acota el valor resultant entre 0 i 1, si el valor és superior a 0,7 el personatge saltarà.[[2]](#footnote-3)

Per entrenar la xarxa neuronal utilitzem una versió d’un algorisme genètic, inspirat en la teoria Darwiniana, anomenat NEAT[[3]](#footnote-4) (Neuro Evolution of Augmenting Topologies). En primer lloc es crearà un nombre N (a definir) de xarxes neuronals, que anomenem de forma individual “agents” i de forma col·lectiva “població” i per cada població l'anomenarem segons el seu ordre d'aparició com a generació. La primera població que apareix serà la generació 0 (zero) i així successivament.

Les connexions de les neurones de cada agent de la generació 0 serà totalment aleatòria i l'estructura de la xarxa neuronal serà predeterminada a una neurona per cada input existent, una neurona de sortida i connexions que connectin les neurones input amb la neurona output. Per la següent generació agafarem el 10% més 1 dels millors agents de la generació anterior, a través d’una puntuació anomenada *fitness* que segons cada tasca té els seus criteris de puntuació, i la resta de població restant es crearà a partir de "mutar" els agents escollits de la generació anterior o de barrejar dues xarxes neuronals que s'han escollit de la població anterior.

La barreja de dos agents, o *crossover (Crs)*, consisteix en comparar si totes dues tenen la mateixa estructura i en aquest cas, es fan les mitjanes aritmètiques de cada connexió amb la seva respectiva per crear un agent nou amb la mateixa estructura i, les mitjanes aritmètiques com els valors de les seves connexions.

Per la mutació d'un agent, el procediment és més complex perquè pot mutar de diferents maneres (la manera s’escollirà de forma aleatòria), de forma no estructural (MNE) o de forma estructural (ME). La no estructural consisteix en canviar un valor d'una connexió per un altre valor aleatori. En canvi, les estructurals - de les que hi ha més d'un tipus - modifiquen el graf perquè poden crear una neurona nova o destruir-la i també perquè poden crear una nova connexió, o destruir-la.

Quan es crea una neurona nova, és necessari connectar-la amb la xarxa, aleshores es crea una connexió que té com a destí aquesta nova neurona i té una altra connexió que parteix des d'aquesta neurona fins un altra, i quan es destrueix una neurona també han de desaparèixer totes les connexions amb les quals interactuava directament.

Un cop s'ha creat la nova generació es repeteix el procés fins a aconseguir un agent que assoleix fer la tasca de forma correcta.

# LA APFLY-IA DINS DEL CONTEXT “*FLAPPY BIRD”*

## FLAPPY BIRD

Imatge que conté captura de pantalla, dibuixos, il·lustració, disseny

Descripció generada automàticament*Flappy Bird* és un joc per dispositiu mòbil que es va popularitzar a partir de l'any 2013 i que consisteix en controlar un ocell que ha de volar entre una sèrie de tubs cilíndrics que apareixen a la pantalla. Per mantenir-lo en vol, el jugador ha de tocar la pantalla enlairant l'ocell lleugerament, i si no es toca, l'ocell cau a causa de la gravetat. L'objectiu principal és passar entre els tubs sense xocar-hi ni caure al terra, i cada cop que l'ocell passa amb èxit entre dos tubs, el jugador guanya un punt.

És un dels videojocs més senzills del qual una IA pot aprendre a jugar. És tracta d’un problema lineal, ja que el seu objectiu és no caure per sota d’una línia horitzontal imaginaria col·locada a la base superior de obstacle, de manera que pugui passar per dins de l’obstacle sense xocar. Aquesta línia és pot aconseguir amb una xarxa neuronal lineal, ja que quan la seva posició en l’eix vertical sigui inferior o igual al de la base de l’obstacle l’agent saltarà.

## FITNESS

Imatge que conté captura de pantalla, text, dibuixos, disseny

Descripció generada automàticamentEl fitness és un sistema de puntuació que s’utilitza per saber quina xarxa neuronal ha realitzat millor la tasca. En la *APFLY-IA* s’entreguen 1000 punts per passar un obstacle, es penalitza amb -100 punts si mor per tocar el terra. Segons com a prop vola l’ocell de la posició Y del centre del forat (PYF) - com és mostra a la imatge - es recompensa l’ocell segons la zona amb 100, 50, 10 o 5 punts per cada segon de més a prop a més llunyà que vola. Finalment per cada segon que l’agent està viu se li entrega un punt. Després de que totes les xarxes neuronals morin o arribin a la puntuació màxima establerta, per fer d’un joc infinit un de finit, després de passar per 150 tubs, és farà una llista de la millor a la pitjor xarxa neuronal segons els criteris anteriors per poder escollir les millors per la següent generació.

## INPUTS

Els inputs són aquelles dades que donem a la xarxa neuronal per que porti a terme la seva tasca correctament. En la APFLY-IA donarem a cada xarxa neuronal 4 possibles dades que anirem variant per poder fer l’anàlisi. Aquestes són: la posició Y de l’ocell (PYO), la posició Y del centre del forat de l’obstacle (PYF), la posició X del centre del forat de l’obstacle (PXF).

## ALTRES CONSTANTS

Per la recollida de dades hi ha variables que canvien, que son els inputs, els tipus de mutació i la població per generació, en canvi n’hi ha d’altres que no varien. Per cada configuració és fan entre 20 i 30 partides, per cada partida hi ha 125 generacions i la puntuació màxima establerta és de 150, es a dir que després d’haver passat per 150 tubs la generació actual morirà i s’iniciarà la següent generació.

# L’APRENENTATGE LOGARITMIC

A l'inici d'aquest treball de recerca una de les hipòtesis plantejades era la forma que prendrien els gràfics que relacionen la puntuació dels agents d'intel·ligència artificial amb el pas les generacions al llarg d'un entrenament.

La nostra hipòtesi era que la forma seria logarítmica, i després de 3.798 partides i més de 450.000 generacions amb diferents paràmetres (població, mutacions i *inputs*) **hem vist que, efectivament, la forma del gràfic demostra que l'aprenentatge és logarítmic, i per confirmar-ho utilitzem un valor anomenat .**

La (coeficient de determinació) en un gràfic logarítmic mesura com de bé s'ajusta una línia de tendència logarítmica a les dades. S'interpreta i calcula de manera similar a la en un gràfic lineal, però considerant la transformació logarítmica. El coeficient varia entre 0 i 1, on 1 suggereix que el model logarítmic explica millor la relació entre les variables i 0 el contrari. Es calcula amb la següent fórmula:

Investigant més a fons podem veure que per molt que sigui logarítmica la forma de la majoria de gràfics, no sempre és logarítmic de la mateixa tendència. A la següent taula s'utilitza el valor mitjà de  en una determinada configuració, relacionant les dades d'entrada i les possibles mutacions per les quals pot estar sotmès la IA sense tenir en compte la població per generació, perquè es va comprovar que la població no modificava la forma del gràfic notablement.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **-** | **PYO** | **PYF** | **PYO, PYF** | **PYO, PYF, PXO** | **PYO, PYF, VYO** | **PYO, PYF, PXO, VYO** | **INPUTS** |
| **-** | 0,535 | 0,21 | 0,228 | 0,638 | 0,759 | 0,764 | 0,608 |  |
| **MNE** | 0,516 | 0,439 | 0,097 | 0,562 | 0,706 | 0,812 | 0,401 |  |
| **ME** | 0,534 | 0,14 | 0,413 | 0,669 | 0,766 | 0,712 | 0,501 |  |
| **MNE, ME** | 0,531 | 0,114 | 0,317 | 0,783 | 0,786 | 0,673 | 0,51 |  |
| **MNE, Crs** | 0,55 | 0,159 | 0 | 0,539 | 0,747 | 0,893 | 0,755 |  |
| **MNE, ME, Crs** | 0,549 | 0,166 | 0,245 | 0,586 | 0,799 | 0,776 | 0,724 |  |
| **MUTACIONS** |  |  |  |  |  |  |  |  |

A la taula es fa referència a les entrades i mutacions de forma numèrica, i aquestes són les corresponents equivalències:

|  |  |
| --- | --- |
| MUTACIONS | ENTRADES |
| “**-**“ 🡪 S'inclouen els valor de totes les possibles configuracions de mutacions.  “**MNE**” 🡪 No estructural  “**ME**” 🡪 Estructural  “**Crs**” 🡪 *Crossover* | “**-**“ 🡪 S'inclouen els valor de totes les possibles configuracions d’entrades.  “**PYO**” 🡪 Posició Y de l’ocell  “**PYF**” 🡪 Posició Y del forat del obstacle  “**PXO**” 🡪 Posició X de l’obstacle  “**VYO**” 🡪 Velocitat Y de l’ocell |

A la taula es pot observar que els aprenentatges menys logarítmics són els de les columnes sense les entrades **PYO** o **PYF**, això es deu al fet que aquestes entrades són imprescindibles perquè la IA pugui aprendre, i per això en aquestes columnes no es produeix cap mena d'aprenentatge. En canvi, la columna on s'utilitzen les entrades **PYO**, **PYF**, **VYO** és on el gràfic té més forma logarítmica i, per ser exactes, la columna on s'utilitzen les entrades **PYO**, **PYF**, **VYO** i les mutacions **MNE**, **Crs**; és on el coeficient és més alt, per tant, és el més logarítmic.

La mitjana del coeficient r^2 dels gràfics que presenten un aprenentatge, durant les 125 generacions, és de 0,73. Aquest valor es indicatiu d’un aprenentatge ràpid logarítmic, però, a que és degut aquest comportament logarítmic?

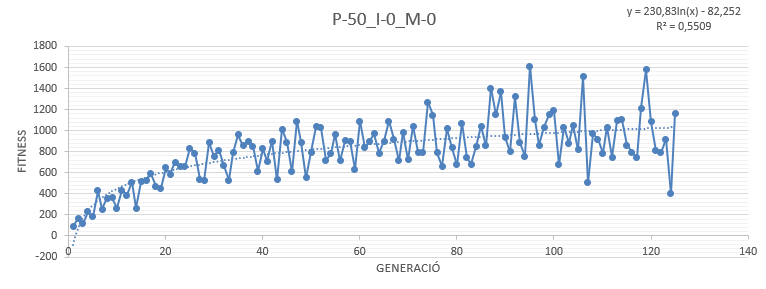
En les primeres generacions, la IA obté guanys significatius en la puntuació de fitness. Això es deu al fet que hi ha moltes possibilitats de millora i, petits canvis en l'estratègia poden portar a grans augments del fitness. De la mateixa manera, a mesura que la IA es troba a generacions mes avançades, els canvis substancials que poden portar a una millora en el fitness són canvis més subtils i la seva millora respecte al fitness també és menor.

Un altre motiu que col·labora a la forma logarítmica és la mecànica del joc. La mecànica del joc del *Flappy Bird* porta a una dificultat finita, de manera que hi ha un “límit” a l’aprenentatge on ja haurà après a tractar la informació de manera 100% correcta.

L'anàlisi d'aquesta tendència logarítmica a priori pot semblar poc útil, però té bastantes aplicacions. Per exemple, permet predir els ràpids guanys inicials de fitness seguits d'un altiplà, de manera que es pot determinar quin és el millor moment per aturar l’aprenentatge o afegir nous reptes a l’aprenentatge per obtenir uns millors resultats.

# COM AFECTEN LES VARIABLES

Les diferents combinacions de inputs, mutacions i poblacions generen diferents comportaments en l’evolució i l’aprenentatge de la xarxa neuronal. A continuació s’explica breument com es comporta la xarxa neuronal depenent de les variables que li introduïm.

ANÀLISI DELS INPUTS

Quan la xarxa neuronal només coneix la posició Y de l'ocell (PYO), els resultats són completament aleatoris anant des de 0 fins a 4.000 de fitness. Mai assoleix una dada superior de 4.000 de fitness, perquè no té la informació necessària per arribar al seu objectiu, degut a que amb les dades que disposa no pot saber a quina posició es troba el forat de l'obstacle. La única cosa que pot fer és evitar caure al terra i estar a una alçada en la qual pot haver-hi un forat. (La puntuació màxima és de 160.000 punts de fitness i una puntuació de 4.000 punts no presenta cap indici d’un aprenentatge rellevant).

Imatge que conté text, captura de pantalla, rebut, línia

Descripció generada automàticamentD’altre banda, quan només li donem la posició Y de l’obstacle (PYF), la xarxa neuronal nomes sap a quina alçada està el forat de l'obstacle i no la posició de l'ocell, per la qual cosa no és capaç de superar-lo ni de manera aleatòria, com sí que passava amb l'input PYO, perquè esta jugant completament a cegues.

Quan la xarxa neuronal coneix la posició Y de l'ocell i la posició Y de l'obstacle, l'aprenentatge de la xarxa és més ràpid i efectiu, perquè té la informació necessària per completar la tasca. Però si ens fixem en les connexions d’aquetes xarxes neuronals ens n’adonarem que la importància del valor de la connexió no és el seu valor en sí mateix sinó en la relació que té amb l’altre connexió.

Imatge que conté text, captura de pantalla, Font, àlgebra

Descripció generada automàticamentPer exemple el quadre que es mostra a continuació, correspon a una xarxa neuronal amb els inputs PYO i PYF on aquest 0,5 podria haver estat un 0,3 o un 0,8 però el que realment importa és que l’altra connexió sigui un -0,5; un -0,3 o un -0,8 respectivament, és a dir que conservin la mateixa relació entre ells.

Imatge que conté text, línia, Font, Trama

Descripció generada automàticamentAleshores podem concloure que amb un sol input que relaciona aquestes dos variables de forma directa, l’aprenentatge és més eficient, millorant la velocitat i reduint la complexitat de la xarxa neuronal, oferint uns millors resultats d’acord amb les gràfiques generació-fitness.

Quan oferim la posició Y de l'ocell i la posició Y i X de l'obstacle (PYO-PYF-PXO) el temps d'aprenentatge augmenta respecta amb la configuració d'inputs (PYO-PYF) degut a l’augment de complexitat de la xarxa neuronal encara que una de les seves opcions sigui ignorar o eliminar aquesta informació també té la possibilitat d’integrar-la a la solució poden resoldre aquest problema de dues diferents.

En canvi, si combinem la posició Y de l'ocell, la posició Y de l'obstacle i la velocitat Y de l'ocell (PYO-PYF-VYO), passa a ser una configuració menys efectiva que la anterior, ja que l'aprenentatge s’ alenteix més per la presència de la velocitat Y, un input que la xarxa ha de ignorar o eliminar.

Imatge que conté text, línia, captura de pantalla, Font

Descripció generada automàticamentQuan ajuntem tots els inputs posició Y de l'ocell, posició Y i X de l'obstacle i velocitat Y de l'ocell (PYO-PFY-PXO-VYO) alenteix encara més l’aprenentatge i passa a ser una de les xarxes neuronals menys optimes, -traient de banda les que no tenen prou informació- degut a que ha d’ignorar el excés d’informació.

És a dir, com més informació innecessària rep la xarxa neuronal, i més complexa és aquesta, l’aprenentatge s’alenteix i/o es limita. Degut a que la xarxa neuronal és lineal i esta pensada per resoldre problemes lineals no massa complexos, com el *Flappy Bird*, l’aprenentatge s’alenteix i/o es limita més que altres xarxes neuronals pensades per resoldre problemes més complexos.

ANÀLISI DE LES MUTACIONS

Imatge que conté text, captura de pantalla, línia, Trama

Descripció generada automàticamentL’aplicació de la mutació no estructural (MNE), en general provoca unes puntuacions de fitness més altes i un aprenentatge més ràpid quan l’apliquem sola, però quan es tracta de generacions on es dona un excés d’informació o on hi falta informació, l’aprenentatge de la mutació s’alenteix degut a que no pot eliminar ni crear connexions i ha de perdre temps aproximant el valor d’una connexió a 0. D’altre banda quan l’ajuntem amb més mutacions perd eficiència (MNE-ME, MNE-Crs i MNE-ME-Crs)).

Imatge que conté text, línia, captura de pantalla, Font

Descripció generada automàticamentQuan apliquem la mutació estructural (ME) només genera un resultat per sobre de les altres mutacions quan es tracta d’aplicar aquesta mutació a generacions amb excés o falta d’inputs. Però la ME és superada per altres mutacions (MNE) al cap d’unes quantes generacions, degut a que la ME inicia amb un primer salt però després s’estanca per fer que la xarxa neuronal sigui més complexa. D’altre banda quan ajuntem la ME amb més mutacions (MEN-ME i MNE-ME-Crs), perd eficiència per generacions amb inputs no gaire bons, però guanya eficiència quan es donen bons inputs d’entrada, assegurant que superi a quan hi ha la mutació sola.

D’altre banda, la implicació del *crossover* no provoca canvis notables en la majoria de partides però a vegades pot fer malbé l’aprenentatge sense cap patró aparent.

En quant a la població, el que fa és augmentar la velocitat d’aprenentatge d’una generació en funció de la població que hi hagi, a causa de la presència de més xarxes neuronals que permeten que aprengui més de pressa.

Cal recalcar que no tots els gràfics permeten corroborar aquestes conclusions al peu de la lletra degut a la quantitat de variables que poden afectar a com es comporta cada generació, però l’anàlisi dels gràfics fets anteriorment busca ser el més precís possible.

# CONCLUSIÓ

Podem concloure que les variables que permetran a la APFLY-IA aprendre de forma més ràpida són els inputs posició Y de l’ocell i posició Y del forat (PYO-PYF), la mutació no estructural i la població més alta que et permetí el dispositiu que utilitzis per entrenar la IA (en el nostre anàlisi la població més alta és de 100).

Com que podem dir que els resultats de l’aprenentatge segueixen la forma d’una funció logarítmica, podem marcar un límit recomanat de generacions per la millor configuració trobada, PYO-PYF, per poder aconseguir la IA que sàpiga realitzar la tasca sense haver de fer les 125 generacions. Segons les dades recollides a partir de la 8ena generació sempre s’obtindrà una IA perfecta (tot i que és possible que en casos extraordinaris encara necessiti alguna generació més).

Som conscients que les possibilitats de nous anàlisis i noves conclusions poden entendres més enllà d’aquest treball i per això deixem a disposició de tothom qui vulgui seguir investigant, el laboratori digital que hem dut a terme en el marc d’aquest treball de recerca. Referencia del Github.

En aquest sentit volem oferir unes primeres línies a futur en algunes de les quals s’està començant a treballar.

# LÍNIES A FUTUR

Mirant cap al futur, la recerca realitzada en aquest treball obre la porta a noves línies d'investigació que podrien ampliar aspectes en els que no s’ha pogut aprofundir o donar-los suficient rellevància, i oferir una continuïtat i ampliació a la recerca fins ara feta.

El primer aspecte a ampliar consistiria en passar d’una xarxa neuronal lineal, plantejada en un inici per resoldre problemes simples, a una xarxa neuronal complexa, i estudiar les diferències entre els dos tipus de xarxes i els resultats obtinguts. Al utilitzar una xarxa neuronal complexa també es podrien plantejar nous tipus de paràmetres i variables a analitzar, la qual cosa faria que els jocs que podrien ser analitzats, a la vegada també podrien ser més complexos. En l’aspecte de l’aprenentatge, també es proposa poder aplicar més d’una mutació per IA i calibrar el nombre d’aquestes per veure si millora algun paràmetre d’aquest aprenentatge.

Per exemple, per fer el problema més complex es podria implementar en el joc una funció en la que aparegués una moneda en una alçada aleatòria i que l’ocell hagués de veure si es viable agafar la moneda o si en agafar-la es moriria i en funció d’això decidir que fer. En aquest sentit s’hauria d’adaptar la xarxa neuronal per determinar la forma correcta d’administrar el fitness i també determinar la millor forma d’entrenar la IA, si per etapes separades per cada dificultat concreta o si directament amb totes les dificultats alhora.

Augmentar les dificultat i sobretot passar d’una xarxa lineal a una complexa implica poder disposar de molta més capacitat de processament de dades (que els equips informàtics dels que disposem fins ara no ens poden proporcionar) per l’obtenció dels resultats. Per agilitar aquest procés s’hauria de millorar la forma de la recollida de dades, ordenant i visualitzant-les d’una forma més senzilla i recopilant-les de forma més rapida i eficient. Per agilitar aquest procés es proposen dos principals camins a seguir:

* Disposar d’equips informàtics més potents i amb molta més capacitat de processament matemàtic (GPU).
* Passar d’un processament i recopilació de les dades de resultats centralitzat a un processament i recollida de dades descentralitzada o distribuïda.

La primera opció és la més immediata i no requeriria pràcticament la modificació dels processos actuals més enllà de les optimitzacions que es creguessin convenients. La principal dificultat d’aquesta opció és el seu cost econòmic i la poca escalabilitat que ofereix a la llarga.

La segona opció, tot i que requereix més esforços de desenvolupament, permetria a un baix cost, obtenir molta més capacitat de processament al repartir de forma distribuïda en diferents sistemes de baixa capacitat la tasca d’obtenció dels resultats.

Per dura a terme aquesta segona opció, s’hauria de repartir, entre quants més col·laboradors voluntaris es pogués, una còpia del joc ja parametritzat amb una paràmetres definits. Els resultats obtinguts es podrien transferir en format JSON (ja implementat al sistema actual) a un sistema central que s’encarregaria de recollir les dades per processar-les, visualitzar-les i analitzar-les en conjunt. Aquest mecanisme es podria fer cíclic per tal d’anar realimentant els jocs amb nous paràmetres a partir dels resultats analitzats.

Tot i afegir més complexitat i requerir més esforç aquesta segona opció permetria a la llarga obtenir més capacitat de processament. Si el projecte busques no només l’anàlisi de resultats sinó també l’aprenentatge de la IA de forma continua, potser la complexitat requerida la podria fer inviable i s’hauria d’optar per la primera opció o una combinació de les dues.

Una darrera línia a futur podria consistir en adaptar la IA del joc en forma de mòdul de tal manera que es pogués utilitzar en altres programes de recerca o directament en jocs dels que es volgués dotar d’una IA per exercí funcions de personatges no jugadors. Per assolir aquest objectiu a banda d’extreure la IA del joc i donar-li forma de mòdul, s’hauria d’establir un estàndard per tal de que el mòdul es pogués acoblar en els sistemes de destí. L’adaptació universal seria molt complexa, però creiem que l’adaptació a plataformes com Godot no hauria de ser excessivament difícil.

# BIBLIOGRAFIA

Glosser.ca. *English: Artificial neural network with layer coloring*. 28 de febrero de 2013. Own work, Derivative of File:Artificial neural network.svg. https://commons.wikimedia.org/w/index.php?curid=24913461 .

McIntyre, Alan, Matt Kallada, Cesar G. Miguel, Carolina Feher de Silva, y Marcio Lobo Netto. «neat-python». Python, 2 de junio de 2024. https://github.com/CodeReclaimers/neat-python.

Neat\_Pyton; *NEAT-Python’s documentation* <https://neat-python.readthedocs.io/en/latest/>

1. Glosser.ca, *English*. [↑](#footnote-ref-2)
2. McIntyre et al., «neat-python». [↑](#footnote-ref-3)
3. NEAT-Pyton [↑](#footnote-ref-4)