Generació procedural de contingut via Reinforcement Learning

Oriol Graupera Serra

Resum — L'objectiu d'aquest projecte és la creació de nivells per al joc de puzles Sokoban mitjançant Reinforcement Learning com a mètode de generació procedural de contingut. Per usar Reinforcement Learning, s'ha representat la generació de nivells com una tasca iterativa on, a cada iteració i mitjançant un sistema de recompenses, es modifica el tauler del joc per obtenir un nivell que tingui una solució. Al llarg del desenvolupament d'aquest treball s'han estudiat diverses estratègies i configuracions en els diferents mòduls del projecte per assolir un sistema capaç de generar nivells de manera estable. Per comprovar la dificultat dels nivells que es creen també s'ha desenvolupat un sistema de validació de la dificultat que relaciona informació extreta del nivell amb un conjunt de nivells etiquetats per dificultat. Els resultats del projecte mostren que el sistema és capaç de generar nivells de manera estable.

Paraules clau — Generació Procedural de Contingut (PCG), Reinforcement Learning (RL), Deep Learning.

Abstract — The aim of this project is to create levels for the puzzle game Sokoban with Reinforcement learning as a Procedural Content Generation method. In order to use Reinforcement Learning, the level generation has been represented as an iterative task where in each step and with a reward system, the Sokoban board is modified to reach a solvable level. During the development of this project, different strategies and configurations have been studied to create Sokoban levels stably. To validate the difficulty of the levels created, a difficulty validation system has been developed, it relates information about the generated levels with a difficulty-labeled dataset of Sokoban levels. The results of the project show that the system is capable of generating solvable levels in a stable way.

Index Terms — Procedural content Generation (PCG), Reinforcement Learning (RL), Deep Learning.

1 Introducció - Context del treball

En aquest treball final d'estudis s'ha desenvolupat un mètode de generació procedural de contingut (en anglès PCG, Procedural Content Generation) que s'encarrega de dissenyar els nivells d'un trencaclosques. Per desenvolupar aquest sistema, s'ha implementat un algorisme de Machine Learning, més concretament, en l'àrea de Reinforcement Learning.

La motivació per a realitzar aquest treball sorgeix de l'interès en el desenvolupament de videojocs més concretament en el món dels trencaclosques i puzles. Des del punt de vista del desenvolupador, la creació i disseny dels nivells per aquests tipus de jocs és una tasca entretinguda i complexa. Quan no hi ha un gran equip darrere, aquesta tasca ocupa una gran quantitat de temps i suposa un punt crític en el desenvolupament. Si a aquest problema li afegim la passió pel machine learning i les ganes d'aprendre a utilitzar noves eines i models, sorgeix la idea de crear un sistema que sigui capaç de crear els nivells d'un videojoc.

Els algorismes de Reinforcement learning utilitzen un mètode iteratiu amb un sistema de recompenses per tal que l'agent aprengui quines són les situacions desitjades [7]. Aquest tipus d'aprenentatge automàtic no fa ús d'un conjunt de dades per entrenar el model i, per tant, pot funcionar adequadament en la generació de nivells d'un joc. En aquests sistemes es pot identificar un agent, que és l'encarregat de generar el contingut que rebrà una recompensa millor o pitjor depenent de la qualitat que tingui. Fent servir aquest valor, es modificarà el contingut per maximitzar el valor de la recompensa fins que s'assoleix l'objectiu desitjat. En aquests algorismes és molt important definir adequadament les recompenses i els objectius, ja que són determinants en el resultat final i l'eficiència del sistema.

El trencaclosques escollit per generar els nivells ha estat un joc japonès anomenat Sokoban. En aquests puzles hi ha un jugador que ha de moure una o més caixes fins a una posició determinada amb el mínim nombre de moviments possible. Per poder moure les caixes el jugador s'ha de col·locar darrere seu i empènyer-les. El jugador només és capaç de moure una caixa cada vegada i aquestes no es poden destruir. La idea del joc és molt senzilla, però depenent del disseny del nivell, pot resultar molt difícil trobar la solució. A la figura 0 es poden observar els moviments necessaris per a solucionar un nivell de Sokoban.

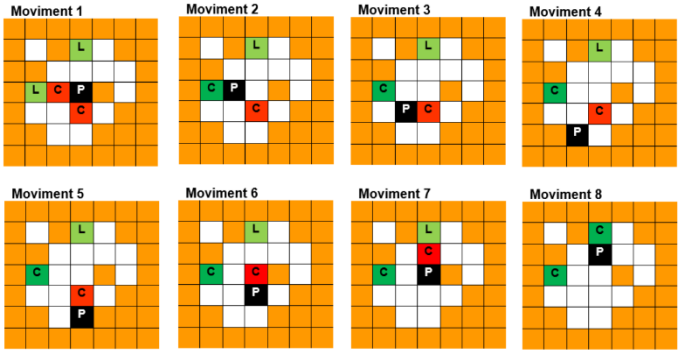

Figura 0. Solució d'un nivell del joc Sokoban.

2 ESTAT DE L'ART

S'entén com a Generació procedural el mètode de crear dades i informació a través d'algorismes. Els algorismes de Generació Procedural s'han fet servir per desenvolupar textures, models 3D, música, històries i altres tipus de contingut relacionat amb els videojocs com podrien ser missions, articles o el disseny del món.

Les tècniques que s'utilitzen per realitzar Generació procedural acostumen a ser algorismes recursius o algorismes de ramificació i poda on el desenvolupador té més control sobre la generació [1]. Un camp que no ha estat molt explorat és l'ús de Machine Learning. Amb aquestes tècniques es poden obtenir resultats molt bons tot i que a vegades la seva utilització pot ser difícil i complexa [4].

Els mètodes d'Aprenentatge supervisat (Supervised Learning) i d'aprenentatge no supervisat (Unsupervised Learning) generalment no són els més utilitzats per a la generació de nivells. Perquè necessiten una gran quantitat de dades per entrenar-se i no hi ha gaires datasets de nivells de jocs disponibles i la seva creació és bastant difícil. Per altra banda, aquests mètodes s'utilitzen com a avaluadors de la qualitat del contingut que s'ha generat. Cal dir que hi ha àmbits de generació de contingut com la generació de gràfics i imatges que sí que tenen moltes dades i per tant funcionen exitosament.

També és comú utilitzar Deep Learning per generar contingut. Encara que aquests mètodes són més complexos i tenen un cost computacional més elevat, són capaços de facilitar la creació de contingut. El Deep Learning permet treballar amb problemes més complexos i obtenir millors resultats. També és comú fer-los servir com a avaluadors del contingut generat per a determinar la qualitat del contingut que es genera [3].

Els algorismes de Reinforcement learning ja han estat utilitzats en vàries ocasions en la creació de contingut per a videojocs. Majoritàriament s'usen perquè la màquina aprengui a jugar al joc, assimilant quines són les regles i normes del joc per arribar a la solució. Com que aquests algorismes no necessiten un conjunt de dades per a l'entrenament, funcionen adequadament amb els jocs. Tot i que no és la seva aplicació més comuna, hi ha diverses investigacions que componen sistemes capaços de crear tota mena de contingut, incloent-hi el disseny de nivells. Les solucions que proposen estan basades a modificar la representació del joc de manera iterativa per aconseguir un nivell que l'usuari pot jugar [5][2].

3 OBJECTIUS

El principal objectiu del treball és desenvolupar un sistema que sigui capaç de dissenyar nivells per al joc Sokoban i que els nivells siguin considerats interessants i prou difícils per a la persona que els juga.

Per tal de crear aquest sistema necessitem els següents subobjectius. Primerament, cal desenvolupar el joc Sokoban conjuntament amb el seu solucionador. Aquest utilitzarà algorismes d'intel·ligència artificial per a trobar el conjunt de moviments òptims que solucionen el nivell. El solucionador és vital per crear el generador de nivells, ja que aquest ha de saber si el nivell que ha generat és solucionable o no i realitzar els canvis necessaris perquè ho sigui.

S'haurà de construir el sistema de Reinforcement learning (RL) adaptant el joc Sokoban perquè funcioni com a entorn de RL, definir i establir les recompenses i penalitzacions així com desenvolupar i entrenar l'agent que aprendrà a generar nivells. Per validar la dificultat dels nivells generats així com per saber si aquests són considerats interessants i complexos es desenvoluparà un algorisme que utilitzarà informació obtinguda del nivell per avaluar-lo.

També s'han definit un conjunt d'objectius secundaris. Per començar es vol que el sistema sigui òptim i eficient i generi els nivells en el menor temps possible. Cal destacar que tant el solucionador com el sistema de validació han de ser eficients i han d'estar optimitzats. També es vol que el sistema a desenvolupar sigui molt adaptable i flexible, perquè es pugui utilitzar per generar nivells per a altres jocs.

4 DESENVOLUPAMENT

En aquesta secció s'expliquen els diferents apartats que s'han desenvolupat en aquest projecte. També s'exposa com estan relacionats entre ells i quins algorismes i estratègies s'han utilitzat.

1	1	1	1	1	1	1
1	0	1	3	0	1	1
1	1	0	4	0	0	1
1	3	2	0	1	0	1
1	0	0	2	0	1	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

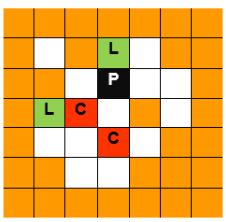

Figura 1. Tauler Sokoban

L'objectiu del joc Sokoban és moure el personatge pel tauler a través dels espais buits per aconseguir col·locar les caixes a les destinacions. En aquest projecte, la mida del tauler és 7x7 i està compost de caselles. Cada casella pot contenir 5 valors diferents que corresponent a paret, caixa, destinació, personatge i espai buit. A la figura 1 es pot veure un exemple d'un nivell de Sokoban i la seva representació numèrica. Com es veu a la figura, les parets són de color taronja (valor 1), els espais buits són blancs (valor 0), les caixes, vermelles (valor 2), les destinacions són de color verd (valor 3) i, per últim el personatge té el color negre (valor 4). S'ha establert que el tauler de Sokoban tingui sempre parets a les caselles més externes, per tant, la mida real del tauler on es treballarà és de 5x5.

Per crear nivells, inicialitzarem el tauler aleatòriament i modificarem les seves caselles de manera iterativa fins que es pugui jugar. Com s'utilitza Reinforcement learning, s'associarà una recompensa al tauler per indicar si la modificació que s'ha fet és positiva o negativa i així aprendre a seleccionar les millors modificacions que ens generen un nivell bo. Per desenvolupar aquest sistema s'han dissenyat diferents mòduls. A la Figura 2 es mostra la composició i relació entre tots ells. Els dos elements més importants són l'Agent i l'entorn de Reinforcement Learning, ambdós estan relacionats entre si i treballen conjuntament en tot moment. L'agent és l'encarregat de prendre una decisió a partir de la recompensa i el tauler actual. Aquests valors són proporcionats per l'Entorn de Reinforcement Learning que és capaç de modificar l'estat del tauler de Sokoban així com calcular la recompensa que té. Per a realitzar aquest càlcul és necessari el Solucionador, el qual és capaç de determinar si un tauler pot arribar a tenir solució i identificar si un nivell és vàlid. El mòdul d'inicialització només s'utilitza a l'inici del procés i determina quin és l'estat inicial del Tauler de Sokoban.

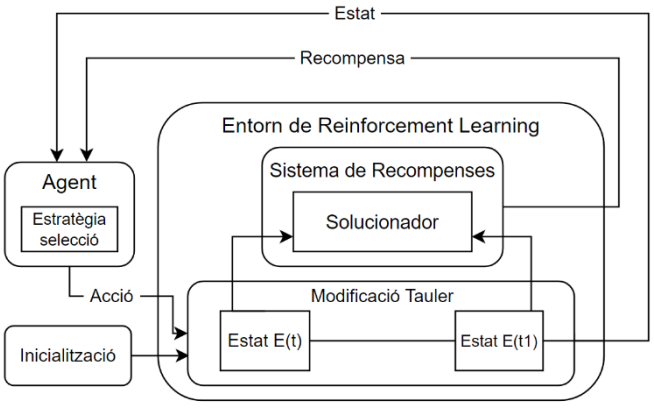

Figura 2. Esquema dels mòduls del sistema desenvolupat.

4.2 Solucionador

El solucionador és el mòdul encarregat de determinar si un nivell és solucionable o no. En aquest mòdul s'ha utilitzat l'algorisme de cerca A*. Es van considerar diferents opcions com Backtracking però només necessitem trobar una solució per categoritzar un nivell com a solucionable o no. A més a més, el problema requereix molta eficiència i un temps d'execució baix, per tant, l'A* s'ajusta perfectament [12].

En l'arbre de cerca de l'algorisme, els nodes estan formats per la posició del personatge i l'estat del tauler. Aquest últim està compost per informació del mateix tauler com per exemple, la localització de les caixes i les destinacions. Cada branca representa els moviments que pot dur a terme el personatge, per tant, la ramificació màxima d'un node és 4, ja que el personatge, en el millor cas pot moure's cap amunt, avall, a la dreta o a l'esquerra. S'han estudiat diferents heurístiques, generalment consisteixen a calcular la distància que hi ha entre un o més elements del tauler. La primera que es va provar va ser la suma de la distància Manhattan entre cada caixa i cada destinació final. Per tenir més precisió es va provar la suma de la distància Manhattan entre cada caixa i la destinació més propera que tenen, aquesta heurística va millorar bastant els resultats obtinguts. En aquests dos casos, no es tenia en compte la posició del personatge així que se'n va desenvolupar una altra que tingues en compte la distància entre el personatge i les caixes a més a més de la distància entre les caixes i la destinació més propera. Contra tot pronòstic, aquesta heurística va empitjorar els resultats de les anteriors. Es van utilitzar altres distàncies diferents de la Manhattan, com l'Euclidiana, però no es van obtenir tants bons resultats ja que no s'ajusten tan bé als moviments que es realitzen en el tauler.

Per millorar el solucionador, es va implementar una detecció d'escenaris que no poden arribar a una solució. L'objectiu és evitar els camins no solucionables i estalviar expandir branques que no arriben a una solució. Aquests escenaris es troben quan una caixa està situada en una posició on el personatge no la pot moure, per exemple en una cantonada. En un tauler de Sokoban hi ha moltes situacions que anul·len la solució i fer una bona detecció és molt important perquè l'algorisme trobi una solució més ràpida. Per últim, també s'ha limitat la dimensió de la cua d'execució de nodes i la profunditat màxima a què arriba l'algorisme. Amb aquestes limitacions es vol evitar perdre molt temps executant l'algorisme i s'assumeix el risc de perdre alguna solució a canvi d'estalviar molt temps i agilitzar el desenvolupament del sistema. Com s'observarà més endavant és molt important tenir un algorisme molt òptim que tingui un temps d'execució baix, ja que és la part del sistema que s'executa més vegades i la que ocupa una major part del temps total d'execució.

4.3 Entorn

L'entorn o enviroment és l'espai de representació del nostre problema de manera que un agent de Reinforcement learning sigui capaç d'interpretar-lo i interaccionar amb ell per aprendre el seu funcionament. En aquest apartat s'explicarà com s'ha definit l'entorn de RL i quines són les característiques que permeten l'aprenentatge d'un agent. Per a desenvolupar l'entorn s'ha utilitzat una llibreria anomenada Open AI Gym [15]. L'objectiu d'aquesta llibreria és facilitar l'entrenament d'agents en entorns de Reinforcement learning així com desenvolupar entorns adaptats a les especificacions del problema en qüestió [10].

L'entorn està format pels següents mòduls: Inicialització, modificació del tauler, sistema de recompenses i l'agent. En aquest treball, l'estat objectiu que es vol assolir, consisteix en crear un nivell solucionable. Amb aquesta premissa, l'agent seleccionarà una acció que modificarà el Tauler, a continuació, per saber si el canvi és positiu o negatiu, es calcularà la Recompensa que indicarà si el tauler

s'apropa o s'allunya a l'objectiu. Aquest procés es repetirà fins que el Tauler arribi a l'estat desitjat. Tots els mòduls estan relacionats entre si; depenent del tipus d'acció, la modificació del tauler realitzarà uns canvis a unes caselles o a unes altres, a més a més, el sistema de recompenses també variarà segons l'estratègia que es segueix. Per últim, les accions també depenen del tipus d'inicialització que s'utilitza. Al llarg d'aquest projecte, s'han desenvolupat diferents solucions per als mòduls de l'entorn, en els següents apartats s'explicarà més detalladament en què consisteixen.

4.3.1 Inicialització de l'entorn

La inicialització de l'entorn consisteix a definir quins són els valors del tauler en el moment de la seva creació. Es considera com el punt de partida des d'on es faran les modificacions necessàries per crear un nivell. Al llarg del desenvolupament del projecte s'han provat dos tipus d'inicialitzacions diferents. En ambdós casos es fa ús de l'aleatorietat per definir els valors de les caselles, ja que es considera una bona tècnica per tenir varietat en la creació de nivells.

Inicialització estàndard: En aquest primer tipus es poden diferenciar dos atributs diferents. En primer lloc, la quantitat de caixes, destinacions, personatges, parets i espais buits que hi ha en el nivell. La major part dels elements del tauler són parets i caselles buides, per tant, es va decidir establir un sistema que fos capaç de col·locar menys caixes, destinacions i personatges i més parets i caselles buides. Per fer-ho s'han definit les probabilitats que té cada tipus de valor d'aparèixer en una casella, com es mostra a la Taula 1. L'altre atribut que es pot identificar consisteix en la disposició de les diferents caselles en el tauler. Es vol evitar tenir un estat inicial amb les caixes molt pròximes a les destinacions o amb nivells que tinguin una distribució similar així que la manera en què es reparteixen els valors en el tauler també és aleatòria. Aquest fet provoca que els estats inicials siguin molt diferents entre ells, augmentant el nombre de possibilitats i diferencies en la creació de nivells.

Tipus	Paret	Buida	Personatge	Caixa	Destinació
Prob.	35%	56%	3%	3%	3%

Taula 1. Probabilitats d'inicialització de les caselles

Inicialització acotada: La primera inicialització no obté bons resultats, ja que el sistema té dificultats per generar nivells solucionables, en la majoria d'iteracions, el nivell queda descartat perquè no conté la mateixa quantitat de caixes i destinacions o el nombre de personatges és superior a un. Aquest segon tipus d'inicialització redueix la complexitat del problema i facilita la creació dels nivells. Així doncs, les caixes i les destinacions es tracten com una parella i s'afegeixen al tauler de manera conjunta. A més a més el nombre de personatges també està limitat a un. Per últim, es considera que hi ha d'haver un mínim i un màxim de parets en el tauler per generar nivells interessants i complexos. Si hi ha molt poques parets, el rang de moviments per al personatge és major i hi ha més solucions. Amb aquesta inicialització s'agilitza el procés i el sistema és capaç de generar nivells de manera més consistent. També podem adaptar el tipus i la complexitat del nivell que volem generar, ja que es pot augmentar o modificar fàcilment el nombre de parets o el nombre de parelles Caixa-Destinació que hi haurà al tauler.

4.3.2 Sistema de recompenses

El sistema de recompenses és l'encarregat de donar una valoració a l'estat del tauler per a saber si aquest s'acosta a l'objectiu o no. Hi ha moltes maneres de desenvolupar les recompenses, utilitzant penalitzacions, recompenses parcials, només valors positius entre tantes altres variacions [11]. Cada sistema de recompenses està molt adaptat a la casuística del problema. Al llarg del desenvolupament del projecte s'han desenvolupat dos sistemes diferents, determinats pel tipus d'inicialització de l'entorn.

En el primer sistema s'utilitzen penalitzacions i recompenses parcials. El valor més baix que es pot assolir és -4 i el valor més elevat és 10. L'ús de penalitzacions està associat als escenaris que contenen més d'un personatge o el nombre de caixes i el nombre de destinacions no és el mateix. En contraposició, les recompenses parcials s'obtenen quan el nombre de personatges, caixes i destinacions és l'adequat. Per poder arribar a la màxima recompensa, s'ha de comprovar si el nivell és solucionable i, per tant, s'executarà el solucionador A*.

Amb la inicialització acotada ens assegurem que el tauler tingui el nombre adequat de caixes, destinacions i personatges, per tant, les penalitzacions i recompenses parcials del primer sistema no són necessàries. En aquest cas 's'utilitza un sistema molt simple on el valor de la recompensa està comprés entre 0 i 1. S'obtindrà el valor més elevat si l'algorisme A* indica que el nivell es pot solucionar, en cas contrari no es rebrà cap recompensa. Tot i la senzillesa del sistema, és capaç de proporcionar bons resultats de manera més ràpida.

4.4 Agent RL

L'agent és l'encarregat de seleccionar quina modificació es realitzarà al tauler a cada iteració fins que s'arribi a l'estat objectiu.

4.4.1 Estratègies de decisió

L'agent té la tasca d'escollir la millor acció d'un conjunt anomenat espai d'accions. L'acció escollida modificarà el tauler per aconseguir que el nivell sigui solucionable. L'espai d'accions està format per les possibles modificacions que se li poden realitzar al tauler i està determinat pel desenvolupador. A l'espai d'accions també se l'anomena estratègia de decisió, ja que les accions que es poden realitzar es consideren com una estratègia i són determinants perquè el sistema pugui crear nivells vàlids. Al llarg del projecte s'han desenvolupat diferents espais d'accions per aconseguir els millors resultats possibles. L'espai d'accions està condicionat pel tipus d'inicialització. En total s'han desenvolupat tres estratègies diferents, les

dues primeres fan servir la inicialització estàndard mentre que l'última utilitza l'acotada.

Estratègia seqüencial: En el primer espai d'accions, l'agent té 5 accions diferents per escollir, aquestes corresponen al tipus de caselles que hi ha al tauler (paret, caixa, personatge, destinació i casella buida). El valor que escull l'agent s'introduirà en una casella del tauler. Cada vegada la casella que es modifica és diferent, però sempre es segueix el mateix ordre, de dreta a esquerra i de dalt a baix de manera cíclica.

Estratègia de control: En el segon espai d'accions, l'agent pot escollir una de les 25 caselles del tauler així com el valor de la casella. L'acció de l'agent estarà composta per les coordenades d'una casella i un valor del 0 al 4. La modificació del tauler consisteix a introduir el valor que ha seleccionat l'agent en la casella escollida. Amb aquest sistema, l'agent té molta més autonomia i control sobre el tauler i és capaç de triar quina és la casella que vol modificar i quin valor s'hi ha d'introduir.

Estratègia d'intercanvi: Amb la inicialització acotada s'ha d'anar amb compte de no alterar el nombre de caselles de cada tipus que hi ha al tauler. Aquesta inicialització ens ajuda a establir els elements necessaris per poder tenir un nivell que es pot solucionar i l'agent ha de mantenir aquests valors del tauler. La seva tasca consistirà a modificar la distribució de les caselles sense canviar el nombre de caixes, destinacions, personatges i parets. L'estratègia de decisió escollida consisteix a realitzar un intercanvi dels valors de dues caselles. L'agent podrà escollir dues de les 25 caselles que hi ha en el tauler i aquestes s'intercanviaran el valor.

4.4.2 Entrenament de l'agent

Per tal que l'agent aprengui quina és la millor acció en cada situació, s'ha d'entrenar. En aquest apartat s'explicarà com funciona l'entrenament de l'agent i les diferents tècniques que s'han utilitzat. Tot i que en un primer moment es va intentar desenvolupar els algorismes de Reinforcement learning de zero [8][9], s'ha decidit utilitzar la llibreria RAY RLlib [13] que té implementats una gran varietat d'algorismes optimitzats i és compatible amb la llibreria Open AI Gym utilitzada pel desenvolupament de l'entorn. A continuació s'explicaran els diferents algorismes que s'han utilitzat per entrenar els agents amb les tres estratègies explicades.

- PPO (Proximal Policy Optimizer): Gradient d'una política (Policy). En cada iteració executa una actualització que minimitza la funció de cost mantenint baixa la desviació amb la política de l'anterior iteració [14].

- DQN (Deep Q-Network): Utilitza xarxes neuronals per a guardar la relació entre l'estat del tauler i la seva recompensa.

- SAC (Soft-Actor Critic): Aquest algorisme optimitza una política (Policy) estocàstica que intenta maximitzar la recompensa i l'entropia.

Com que les estratègies de decisió tenen uns valors de sortida diferents, no tots els algorismes es poden utilitzar en totes les estratègies. Per exemple DQN només es pot fer servir amb l'estratègia seqüencial, ja que aquest algorisme només té com a sortida un valor. L'algorisme PPO és molt flexible i es pot utilitzar arreu mentre que SAC només s'utilitza amb l'estratègia d'intercanvi.

L'entrenament de l'agent funciona de la següent manera: en cada iteració d'entrenament, es generen 100 nivells i per a cada nivell es permetrà realitzar fins a 100 accions. Cada vegada que es realitzi una acció, s'ajustaran els pesos del sistema amb l'ajuda de la recompensa. El nombre total d'iteracions d'entrenament que s'han realitzat per a cada agent és de 2000.

5 VALIDACIÓ DE LA DIFICULTAT

Una part important d'aquest projecte consisteix a saber quina dificultat tenen els nivells que es generen. La manera més comuna de determinar la dificultat és utilitzar els comentaris de jugadors que han jugat al nivell que s'està avaluant. Aquest enfocament necessita un gran conjunt de persones que indiqui la dificultat del nivell. Tot i que els resultats que es poden obtenir són bastant precisos, no sempre és la millor opció. En aquest cas, s'ha desestimat i s'ha escollit extreure informació del tauler per definir quins elements són més característics dels nivells difícils i dels fàcils. D'aquesta manera es pot relacionar la informació extreta del tauler amb la dificultat del nivell.

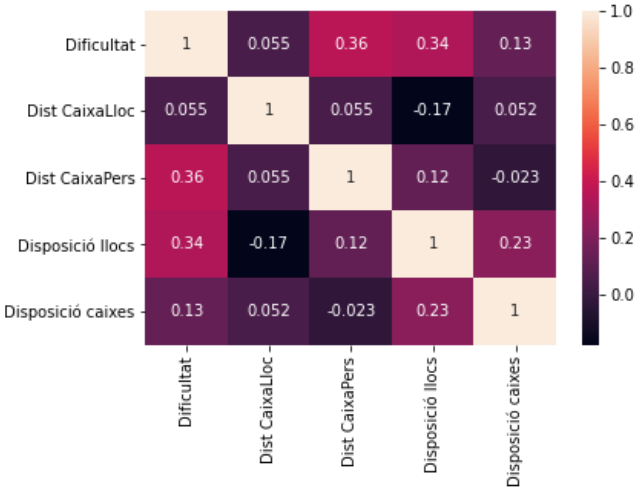

Figura 3. Heatmap de la relació entre la dificultat i la informació del tauler.

Per tal d'establir aquesta relació és necessari utilitzar un conjunt de dades de nivells de Sokoban etiquetats amb la dificultat. No hi ha gaires conjunts de dades d'aquest trencaclosques i encara menys que s'adaptin a les caracte-

rístiques del problema. Tot i les complicacions, s'ha trobat un dataset que divideix els nivells en dues dificultats i conté 3000 nivells de la dificultat més elevada i 40000 de dificultat mitjana [19]. Encara que aquest conjunt de dades és el que s'adapta millor al problema, es considera necessari tenir més de dos dificultats per ser capaços d'avaluar correctament els nivells generats.

Per completar el dataset es va decidir afegir nivells de dificultat molt fàcil creats pel generador desenvolupat. Amb la inicialització acotada som capaços d'adaptar la creació del nivell i si aquests només contenen una parella caixa-destinació i el nombre de parets és baix, els nivells seran molt simples i fàcils de solucionar. Amb aquesta configuració s'han generat 3000 nivells que s'han afegit a la resta. Perquè el conjunt de dades estigui balancejat s'han utilitzat 3000 nivells de cada tipus de dificultat, que sumen un total de 9000. Amb el dataset completat només falta escollir quina informació del tauler es considera determinant per a la dificultat del nivell. En total s'han seleccionat 10 atributs diferents:

- El numero de parelles Caixa Destinació que hi ha en el tauler .
- La longitud del camí solució.
- La distància entre les caixes i les seves destinacions (Dist CaixaLLoc).
- La distància entre el personatge i les caixes (Dist CaixaPers).
- La disponibilitat de les destinacions, és a dir, quantes parets tenen al voltant (Disposició llocs).
- La disponibilitat d'accés de les caixes (Disposició caixes).
- La quantitat de moviments en direcció a les destinacions que té el camí solució.
- La quantitat de moviments en direcció oposada a les destinacions que té el camí solució.
- Congestió del camí entre caixa i destinació
- La mida del mapa

Per determinar la relació entre aquests elements i la dificultat, s'han utilitzat els 9000 nivells etiquetats del dataset. Per a cada nivell s'han identificat els 10 elements i s'ha calculat la correlació que tenen amb la dificultat etiquetada del nivell. Cal mencionar que alguns elements no s'han pogut utilitzar, ja que no s'ha aconseguit extreure la informació per a tots els nivells. El solucionador utilitzat té dificultats per solucionar els nivells etiquetats com a dificultat elevada i, per tant, els elements que l'utilitzen com per exemple la longitud del camí solució o la quantitat de moviments en direcció les destinacions, no es poden utilitzar. A més a més, els nivells generats pel sistema només tenen una o dues caixes mentre que els del dataset existent, en tenen 4, per tant, la relació entre el nombre de parelles Caixa-Destinació que hi ha en el tauler i la dificultat no es considera vàlida i no es pot utilitzar. La mida del mapa també és diferent per als nivells generats (5x5) i els del dataset (8x8) i, per tant, tampoc se'n pot fer ús.

A la Figura 3 es pot veure el mapa de calor que mostra la correlació entre els elements utilitzats i la dificultat. S'observa que moltes variables no estan gens relacionades amb la dificultat i, per tant, no es poden utilitzar per definir si un nivell és fàcil o difícil. Hi ha dos elements que si mostren una certa relació amb la complexitat dels nivells, aquests són la distància entre el personatge i les caixes i la distància entre les caixes i les destinacions, aquest valor no és suficientment alt per valorar la dificultat dels nivells.

És important mencionar que la tasca de determinar la dificultat d'un puzle és molt més complexa del que s'esperava. El fet d'identificar els elements que fan que un nivell sigui més complicat pot resultar ser una tasca molt laberíntica i entretinguda [16][18]. A més a més la dificultat és una característica que els humans donem a un nivell, les màquines no tenen la mateixa percepció que nosaltres i interpreten de manera diferent el tauler. La validació de la dificultat desenvolupada no es considera òptima i es creu que s'hauria d'utilitzar un conjunt de dades més extens i complet amb més varietat de dificultats i diversitat en els nivells per considerar la correlació entre els elements més robusta i fiable. També és important tenir la capacitat de resoldre tots els nivells del dataset per poder utilitzar tota la informació que s'ha extret del tauler.

En definitiva, aquest sistema de validació no compleix amb les expectatives proposades i s'ha arribat a la conclusió que amb el conjunt de dades creat no som capaços de determinar la dificultat dels nivells generats de manera estable i adequada.

6 RESULTATS

Aquest apartat es divideix en dues seccions, en la primera s'explicaran els resultats de l'entrenament dels agents mentre que en la segona es mostraran els resultats de la generació de nivells.

6.1 Entrenament

A la figura 4 es poden veure els resultats de l'entrenament utilitzant l'estratègia de decisió seqüencial amb els algorismes PPO i DQN. Com es pot observar a la gràfica, l'agent que utilitza PPO és capaç de millorar la recompensa obtinguda en les primeres iteracions de l'entrenament. A partir de la iteració 250 es queda estancat i no té cap creixement destacable. Pel que fa a DQN, els valors de la recompensa són molt dispersos i inestables, en alguna iteració assoleix una recompensa elevada però és incapaç de mantenir-la. Amb aquesta primera estratègia els resultats no són gaire bons, la millor recompensa que pot assolir el sistema és de 10 i en cap dels dos algorismes sobrepassa un valor de -1. El sistema no aconsegueix crear un nivell que només tingui un personatge i que tingui el mateix nombre de caixes i destinacions,

aquest aprenentatge és imprescindible per a poder crear nivells. Es considera que amb aquesta estratègia l'agent està molt limitat, ja que les caselles es modifiquen en un ordre en concret i l'agent només pot escollir el valor de la casella.

Sobre l'estratègia de control només s'ha utilitzat l'algorisme PPO, els resultats de l'entrenament es poden veure a la figura 5. En aquest cas l'agent també mostra un aprenentatge progressiu fins la iteració 750, a partir de llavors no mostra un ascens important. Aquesta estratègia deixa tot el control a l'agent. Aquest fet suposa que l'agent tingui més dificultats per aprendre el funcionament de l'entorn però es considera que, a llarg termini, els resultats poden arribar a ser molt positius. Aquest agent té moltes capacitats i s'hauria d'entrenar més iteracions per tenir una millora important.

En el cas de l'estratègia d'intercanvi es pot modificar la inicialització del tauler. S'ha observat que aquesta té una relació directa amb la capacitat de generar nivells. Amb la inicialització acotada es poden definir la quantitat de parets i parelles caixa-destinació. Si reduïm aquests valors, serà més fàcil crear un nivell solucionable però com a conseqüència aquest serà més fàcil i senzill. S'ha entrenat l'agent amb la següent configuració: només es permet tenir fins a dues parelles caixa-destinació i 10 parets al tauler, aquesta configuració implica que els nivells que es generen no tinguin una complexitat molt elevada. A la figura 6 es poden veure els resultats obtinguts amb els algorismes PPO i SAC. Com es pot veure en la figura l'agent que utilitza PPO és capaç d'obtenir una recompensa bastant elevada però no mostra cap aprenentatge i s'obté un valor bastant constant. Per altra banda, l'agent SAC, no assimila el funcionament de l'entorn i obté uns valors molt baixos. Per les característiques d'aquest problema i els resultats obtinguts es considera que l'agent SAC no s'adapta correctament i és incapaç d'aprendre.

Amb la inicialització acotada es poden generar nivells de manera consistent i millorar els resultats de les altres estratègies però l'agent no mostra cap tipus d'aprenentatge. Es considera necessari entrenar molt més els agents per observar l'efectivitat de l'estratègia d'intercanvi.

Amb la inicialització estàndard, l'agent té més marge d'aprenentatge ja que primer s'ha d'enfocar en aconseguir una distribució que contingui un personatge i el mateix nombre de caixes i destinacions per després aprendre com s'han de col·locar les caselles perquè el nivell sigui solucionable. De fet, amb les estratègies seqüencial i de control, l'aprenentatge que tenen els agents està enfocat en la tasca d'aconseguir la distribució comentada. L'estratègia d'intercanvi, només pot aprendre a realitzar la segona tasca de col·locació de caselles que és molt més complexa i com es mostra a les gràfiques, no s'aconsegueix.

Figura 4. Entrenament estratègia seqüencial

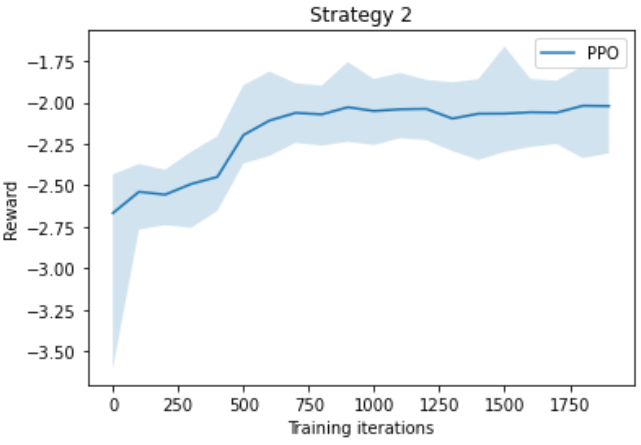

Figura 5. Entrenament estratègia de control.

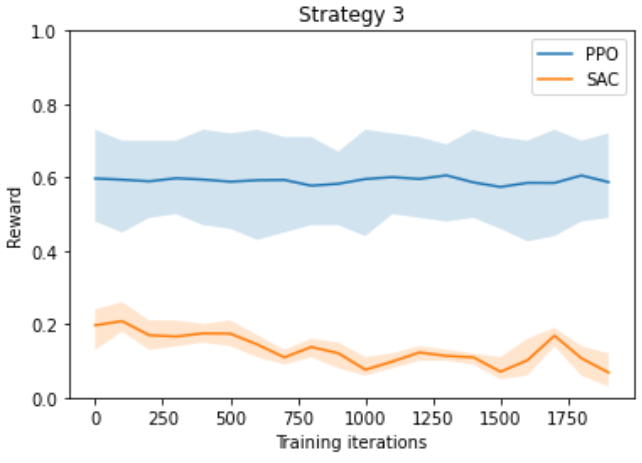

Figura 6. Entrenament estratègia d'intercanvi.

6.2 Generació de nivells

En aquest apartat es mostraran les capacitats del sistema desenvolupat, fent èmfasis en la creació de nivells solucionables. Per poder avaluar i comparar la generació de nivells, s'han utilitzat les estratègies de control i d'intercanvi amb els algorismes PPO i SAC així com amb una selecció de l'acció aleatòria (Random). Els nivells generats mitjançant l'estratègia d'intercanvi tenen com a

configuració de la inicialització acotada un màxim de deu parets i dues parelles caixa-destinació. S'ha desestimat comparar els resultats amb l'estratègia seqüencial perquè no és capaç de generar nivells.

Per a avaluar la creació de nivells solucionables, es generaran 10 000 nivells per a cada estratègia diferent. Sobre aquesta generació s'extrauran les següents mètriques: En primer lloc, el percentatge de nivells que es poden solucionar, aquesta dada ens indica com de bo és el nostre generador. Com més elevat sigui el percentatge, més nivells solucionables generarà i, per tant serà més eficient. En segon lloc, també es comptabilitzaran la quantitat de vagades que l'agent escull una acció per a generar un nivell solucionable. Com que en total s'intentaran generar 10 000 nivells, es calcularà la mediana d'accions necessàries per obtenir un nivell amb solució. A continuació també es comptaran les regions que tenen els taulers creats. Les regions són els diferents espais de moviment que hi ha en el tauler, zones que estan separades per parets i el personatge només hi pot accedir si ja es troba a dins. L'estat ideal només conté una regió en el tauler, ja que si n'hi ha més d'una, hi ha zones que són inaccessibles per al personatge. En aquest cas es calcularà la mitjana de regions que hi ha en els taulers solucionables. Per últim, també calcularem la mitjana de parelles Caixa-Destinació que és generen en els 10 000 nivells. D'aquesta manera podem validar la capacitat de produir nivells, i la seva qualitat.

Per als 10 000 nivells s'ha limitat la quantitat d'accions que permetem realitzar a l'agent en cada nivell. Els nombres d'accions que s'han analitzat són 150, 100 i 50. Aquests valors posen a prova als agents, ja que en l'entrenament estan limitats a 100 accions per nivell. Com més elevat és aquest valor, més fàcil és generar un nivell, ja que es realitzen més modificacions que poden produir un nivell bo. Un dels objectius del projecte consisteix en produir els nivells de manera ràpida i eficaç i per tant volem ser capaços de crear els nivells en el menor nombre d'accions possibles.

Agent	%Nivells Solucionables	Mediana Accions per nivell	Mitjana Regions	Mitjana Caixa-Destinació
PPO Control	0.4%	1	1.7	1
RANDOM Control	0.15%	1	1.86	1
SAC Intercanvi	10.6%	1	1.58	1.2
PPO Intercanvi	66.67%	34	1.66	1.43
RANDOM Intercanvi	68.77%	35	1.64	1.45

Taula 2. Generació de 10 000 nivells, 150 accions per nivell

Agent	%Nivells Solucionables	Mediana Accions per nivell	Mitjana Regions	Mitjana Caixa-Destinació
PPO Control	0.21%	1	1.74	1
RANDOM Control	0.16%	1	1.86	1
SAC Intercanvi	11.28%	1	1.58	1.2
PPO Intercanvi	57.69%	27	1.63	1.4
RANDOM Intercanvi	58.69%	28	1.64	1.39

Taula 3. Generació de 10 000 nivells, 100 accions per nivell

Agent	%Nivells Solucionables	Mediana Accions per nivell	Mitjana Regions	Mitjana Caixa-Destinació
PPO Control	0.25%	1	1.52	1
RANDOM Control	0.2%	1	1.86	1
SAC Intercanvi	9.89%	1	1.59	1.2
PPO Intercanvi	40.75%	14	1.62	1.31
RANDOM Intercanvi	41.53%	16	1.61	1.33

Taula 4. Generació de 10 000 nivells, 50 accions per nivell

A la taula 2 es poden observar els resultats limitant el nombre d'accions a 150. Pel que fa a l'estratègia de control, tant l'agent aleatori com el PPO només són capaços de generar nivells amb una parella caixa-Destinació. Observem que la mediana d'accions té un valor d'1 que ens indica que els nivells generats són fruit de l'aleatorietat. Aquests resultats es repeteixen en les taules 3 i 4 on el nombre d'accions està limitat a 100 i 50 respectivament i arreu s'obtenen uns valors similars als de la taula 2. Pel que fa al percentatge de nivells solucionables, l'agent PPO mostra una lleu millora respecte l'agent Random, aquesta millora s'atribueix a l'aprenentatge de l'agent en la tasca de col·locar la mateixa quantitat de destinacions i caixes així com de posar tan sols un personatge en el tauler. Com s'ha comentat en els resultats de l'entrenament, i s'ha demostrat en les taules, l'estratègia de control no és capaç de generar nivells.

Per altra banda, l'agent SAC amb l'estratègia d'intercanvi també té dificultats per obtenir bons resultats. El percentatge de nivells solucionables que genera està al voltant del 10%, sense importar el límit d'accions establert. La mediana d'accions utilitzades per crear un nivell amb solució té un valor d'1, per tant aquest agent, igual que els agents amb l'estratègia de control, també depèn de l'aleatorietat per generar nivells. Les decisions que pren l'agent no tenen importància en la creació de nivells solucionables i per tant, l'agent no ha après el funcionament

del sistema. En aquest cas, els resultats són millors que els dels agents amb l'estratègia de control perquè s'utilitza la inicialització acotada.

Per últim, els dos agents que han obtingut millors resultats i són capaços de crear nivells de manera consistent amb les diferents limitacions del nombre d'accions són l'agent Random i el PPO amb l'estratègia d'intercanvi. Com es preveia, s'obté el major percentatge de nivells solucionables al limitar el nombre d'accions a 150. Com es veu a la Taula 2, gairebé el 70% dels nivells són solucionables. Els dos agents, Random i PPO, obtenen uns resultats molt similars en totes les mètriques, aquest fet confirma els resultats de l'entrenament, ja que l'agent PPO, tot i ser capaç de crear nivells amb solució, no ha après la tasca de generació de nivells i els resultats són semblants a la selecció aleatòria de les accions (Agent Random). Amb aquests agents s'aprecia que la mediana d'accions augmenta com més accions es permeten realitzar per nivell, el valor més elevat és de 34 accions amb la limitació de 150. Respecte al nombre mitja de regions dels nivells amb solució es manté constant en les diferents taules. Aquests resultats són positius i demostren que encara que es realitzin moltes accions en un nivell, la quantitat de regions no es veu afectada. En el cas del valor mig de parelles caixa-destinació, com més accions es permeten realitzar, més parelles per nivell hi ha, aquest fet indica que l'agent, no es limita a crear nivells amb tan sols una parella caixa-destinació i també és capaç de generar nivells més complexos. Com s'observa a les Taules 2, 3 i 4, com més elevat és el nombre d'accions permès (150, 100 i 50), major és el percentatge de nivells solucionables 68%, 58% i 41% respectivament. Aquest esdeveniment indica que la generació de nivells solucionables depèn de la quantitat d'accions que es realitzen, com més se'n fan, més fàcil és crear un nivell que tingui solució. Amb aquests agents també s'han intentat generar 10 000 nivells amb una limitació de 500 accions, en aquest cas, més del 90% dels nivells són solucionables.

6 Conclusions

En aquest projecte s'ha aconseguit crear uns sistema capaç de generar nivells per al joc de puzles Sokoban. Per fer-ho s'han desenvolupat diferents mòduls connectats entre si que componen un entorn de Reinforcement Learning i permeten l'entrenament d'un agent. En aquest projecte s'han estudiat diferents tipus d'agents, estratègies de decisió, inicialitzacions del tauler i sistemes de recompenses per valorar el seu impacte en la creació de nivells. Es considera que els agents utilitzats necessiten més iteracions d'entrenament perquè siguin capaços d'entendre el funcionament de l'entorn i aprenguin a generar nivells solucionables. Tot i la falta d'entrenament, s'ha aconseguit generar nivells de manera estable utilitzant l'estratègia d'intercanvi. També tenim la capacitat de controlar la dificultat dels nivells que es generen a través de la inicialització del tauler.

Les millores del treball consisteixen en primer lloc, en optimitzar el solucionador A*. Aquest mòdul és el més executat del sistema, per tant és important que sigui ràpid i eficient. A més a més hauria de ser capaç de solucionar nivells més complexos, la dificultat dels nivells que es generen depèn, entre d'altres, de les aptituds del solucionador, en alguna ocasió es generen nivells molt complexos que el solucionador no és capaç de resoldre. Per altra banda, també és important entrenar durant més iteracions els agents, d'aquesta manera es podrà valorar i analitzar millor el seu aprenentatge. Pel que fa a les estratègies de decisió, també seria interessant estudiar-ne diferents per comparar els resultats i avaluar l'agent. En la validació de la dificultat, es necessari crear un conjunt de dades molt més complert i establir un sistema capaç de relacionar la informació del tauler amb la dificultat dels nivells de manera òptima. També seria molt interessant incloure en el sistema de recompenses la dificultat del nivell, d'aquesta manera es podria controlar la dificultat dels nivells que es generen.

Per últim, comentar que la magnitud del problema plantejat ha estat més elevada del que s'esperava, en aquest sistema hi ha molts mòduls diferents i la major part tenen una gran complexitat. Tot i així es fa un balanç positiu del desenvolupament d'aquest projecte on s'han après tècniques i eines que s'utilitzen en Reinforcement learning així com el funcionament d'aquest tipus d'aprenentatge computacional. També s'ha aconseguit adaptar el problema de la generació de nivells en una tasca de Reinforcement Learning a sobre d'analitzar i estudiar diferents estratègies per la resolució del problema.

Bibliografia

[1] Blomberg, J., Jemth, R., Lennar, A., Lilius-Lundmark, R., Pettersson Johnsson, M., & Svensson, T. (2018). *An Exploration of Procedural Content Generation for Top-Down Level Design* (Bachelor's thesis).

[2] Bontrager, P., & Togelius, J. (2021, August). Learning to Generate Levels From Nothing. In *2021 IEEE Conference on Games (CoG)* (pp. 1-8). IEEE.

[3] Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G. N., & Togelius, J. (2021). Deep learning for procedural content generation. *Neural Computing and Applications*, *33*(1), 19-37.

[4] Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., ... & Togelius, J. (2018). Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games*, *10*(3), 257-270.

[5] Khalifa, A., Bontrager, P., Earle, S., & Togelius, J. (2020, October). Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Vol. 16, No. 1, pp. 95-101).

[6] Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172-186.

[7] Moni, R. (SmartLab AI). (2019). Reinforcement learning algorithms – An intuitive overview. Web link: https://medium.com/@SmartLabAI/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc

[8] Kansal, S.(2018). Reinforcement q-learning from scratch in python with openai gym, https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

[9] Torres, J. (2021). Introducción al aprendizaje por refuerzo profundo: Teoría y pràctica en Python. Independently published

[10] Ayoosh Kathuria, (2021) Getting started with Open AI Gym. https://blog.paperspace.com/creating-custom-environments-openai-gym/

[11] Bonsai, (2017), Writing Rewards Functions. https://medium.com/@BonsaiAI/deep-reinforcement-learning-models-tips-tricks-for-writing-reward-functions-a84fe525e8e0

[12] Junghanns, A. (2000). Pushing the limits: new developments in single-agent search. University of Alberta.

[13] RAY, RLlib: Industry-Grade Reinforcement Learning. https://docs.ray.io/en/latest/rllib/

[14] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[15] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[16] Jarušek, P., & Pelánek, R. (2011, March). What Determines Difficulty of Transport Puzzles?. In *Twenty-Fourth International FLAIRS Conference*.

[17] Kartal, B., Sohre, N., & Guy, S. J. (2016, September). Data driven Sokoban puzzle generation with Monte Carlo tree search. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[18] Jarušek, P., & Pelánek, R. (2010). Difficulty rating of sokoban puzzle. In *STAIRS 2010* (pp. 140-150). IOS Press.

[19] Guez, A., Mirza, M., Gregor, K., Kabra, R., Racanière, S., Weber, T., ... & Lillicrap, T. (2019, May). An investigation of model-free planning. In *International Conference on Machine Learning* (pp. 2464-2473). PMLR.

[20] Jarušek, P., & Pelánek, R. (2010). Human problem solving: Sokoban case study. *Technická zpráva, Fakulta informatiky, Masarykova univerzita, Brno*.

[21] Li, X., Li, L., Gao, J., He, X., Chen, J., Deng, L., & He, J. (2015). Recurrent reinforcement learning: a hybrid approach. *arXiv preprint arXiv:1509.03044*.