Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

FIB

Grau en Enginyeria Informàtica, menció en Computació

Sistema recomanador per la formació d'equips utilitzant *reinforcement learning*

Autor: David Valero Masachs
Director: Prof. Dr. Cecilio Angulo Bahón

Treball de Final de Grau

Barcelona, 18 de Gener 2021

Resum

En el món universitari cada vegada és més freqüent que els estudiants hagin de fer pràctiques en una empresa durant el seu grau universitari, així i tot no hi ha sistemes que simplifiquin la feina d'assignació dels estudiants als diferents projectes oferts per les empreses.

Aquest treball final de grau presenta, mitjançant l'ús d'algorismes de *reinforcement learning* (aprenentatge per reforç), una eina que dóna als estudiants tantes alternatives com es vulgui segons les seves preferències, els projectes que s'ofereixen i els perfils d'estudiants que volen cadascuna de les empreses per cada projecte. A més, també permet que cada estudiant pugui ordenar-les al seu gust i el sistema, intentant satisfer al màxim possible als estudiants, dóna una assignació final.

Per dur a terme aquest projecte s'ha utilitzat Python com a llenguatge de programació i la llibreria OpenAI per facilitar la creació de l'entorn. A causa de les característiques del problema, per l'agent s'ha utilitzat la llibreria Stable Baselines i l'algorisme d'aprenentatge per reforç A2C per realitzar la primera assignació.

L'assignació final també utilitza la mateixa estructura i algorisme, però per facilitar l'entrenament, un bucle assigna prèviament tots els estudiants que pot assignar directament i l'agent A2C és l'encarregat d'assignar els que no se'ls hi ha pogut assignar cap projecte de la seva llista inicial.

Abstract

In the university world it is increasingly common for students to do internships in a company during their university degree, but there are no systems that simplify the work of assigning students to the different projects offered by companies.

This final degree project presents, through the use of *reinforcement learning* algorithms, a tool that gives students as many alternatives as they want according to their preferences, the projects that are offered and the student profiles that each of the companies wants for each project. In addition, it also allows each student to order their alternatives to their liking and the system, trying to satisfy as many students as possible, gives a final assignment.

To carry out this project Python has been used as a programming language and OpenAI library to facilitate the creation of the environment. Due to the characteristics of the problem, the agent used Stable Baselines library and A2C reinforcement learning algorithm to perform the first assignment.

The final assignment also uses the same structure and algorithm, but to facilitate training, a loop pre-assigns all the students it can assign directly. The A2C agent is responsible for assigning students who cannot be assigned to any project on their initial list of alternatives.

Agraïments

Aquest treball no l'hauria pogut dur a terme sense l'ajuda del director Prof. Dr. Cecilio Angulo Bahon, que ha estat disponible en tot moment des del primer dia per resoldre qualsevol dubte i donar-me una confiança extra en tot el camp del Reinforcement Learning que desconeixia.

També m'agradaria donar les gràcies a la meva família i amics, que amb el seu suport i ànims m'han donat forces extres per continuar en els moments més difícils i per poder acabar el projecte.

Continguts

Llista de figures

Llista de taules

Llista de llistes

Llista d'abreviacions

UPC	Universitat Politècnica de Catalunya
FIB	Facultat d'Informàtica de Barcelona
ETSEIB	Escola Técnica Superior d'Inginyeria Industrial de Barcelona
TFG	Treball de Final de Grau
TFM	Treball de Final de Master
IA	Inteligència Artificial
RL	Reinforcement Learning
RH	Recursos Humans

1. Contextualització i abast del projecte

1.1 Context

Aquest és un Treball de Final de Grau en Enginyeria Informàtica, menció en Computació, realitzat a la Facultat d'Informàtica de Barcelona de la Universitat Politècnica de Catalunya, dirigit pel Prof. Dr. Cecilio Angulo Bahon.

1.1.1 Introducció

Un cop decidit quin Grau Universitari fer, l'estudiant entra en un període on tot sembla conduït. Com a molt triar assignatures i quines activitats fer. Però, tot canvia quan vol fer pràctiques. Moltíssimes opcions, de disciplines que encara no sap si li agraden o no, a empreses que coneix només pel nom i pel sector on són... Molts són els elements a tenir en compte i molt poc el suport disponible per escollir.

En el context educatiu, cada vegada és més habitual que els estudiants hagin de fer treballs pràctics en una empresa com a part del seu currículum, ja que s'ha comprovat que l'ocupabilitat dels estudiants al final dels estudis augmenta gràcies a aquestes pràctiques.

A més, les empreses es beneficien d'aquest programa, ja que obtenen estudiants motivats que treballaran per salaris reduïts mentre reben una formació. Per la seva part, els estudiants es beneficiaran d'un primer contacte amb el mercat laboral.

Actualment, els centres educatius fan les assignacions de pràctiques a mà, amb totes les limitacions que això comporta: molt de temps invertit, molts estudiants i empreses que no acaben satisfets amb l'assignació... Des de la UPC sorgeix l'interès a crear una eina que ajudi els estudiants a escollir on fer les pràctiques curriculars dels estudis d'una manera transparent i ràpida. A més a més, es satisfan les necessitats de les empreses que busquen nou talent.

1.1.2 Problema a resoldre

L'objectiu d'aquest treball és crear una eina per centres educatius, estudiants i empreses que ajudi als estudiants a triar quines pràctiques volen fer.

La intenció és que l'eina generi mitjançant un algorisme una solució que assigni totes les places disponibles per fer pràctiques als diferents estudiants que en busquen. Aquesta solució ha de ser la que satisfaci millor totes les condicions que estudiants i empreses han facilitat prèviament mitjançant un qüestionari.

Perquè els i les estudiants tinguin opinió en la solució final, la intenció és que l'eina, en comptes de facilitar només una única opció a cada estudiant, li doni la possibilitat de triar entre dues o més pràctiques.

Una vegada els estudiants han ordenat en ordre de preferència les opcions proposades, l'eina ha de generar una solució que satisfaci de la millor manera possible les eleccions que han fet els estudiants.

1.1.3 Stakeholders

Tenim diferents actors dins del treball, els quals els podem agrupar en dos grups depenent de la interacció que tinguin amb el projecte.

Primerament, els stakeholders que tenen interacció directa en la realització del treball són el tutor i l'investigador.

El professor Cecilio Angulo Bahon és el director del treball, dirigirà i guiarà l'investigador per al correcte desenvolupament del projecte.

L'investigador, en David Valero Masachs, és l'encarregat de planificar, desenvolupar i documentar el projecte, a més d'experimentar, analitzar i treure conclusions.

En segon lloc, els grups d'interès que no interactuen amb el projecte, però reben beneficis directes es poden dividir en quatre grups més:

- Estudiants que han d'escollir pràctiques i als que se'ls assignarà una plaça dins de totes les opcions possibles.
- Professorat responsable de guiar a l'estudiant en l'elecció i facilitar-li informació i contactes a les empreses.
- Tècnics especialistes de les empreses que volen nou talent, responsables de parlar amb el professorat sobre quin tipus de persona necessita.
- Empreses necessitades de nou talent pels seus projectes.

1.2 Justificació

Aquesta temàtica ha sigut recurrent en els últims anys a causa de la creixent utilització de sistemes d'Intel·ligència Artificial (IA) per resoldre tota mena de problemes de la vida diària.

Des de l'Institut d'Investigació en Intel·ligència Artificial (IIIA - CSIC) van presentar en un workshop de la *International Conference On Autonomous Agents and Multi-Agent Systems* (AAMAS) d'enguany l'algorisme *TAIP* [1], revisat amb el nom d'*Edu2Com* [2] durant l'*AI for Social Good 2020*. Aquest algorisme és un algorisme heurístic que genera una assignació 'a un pas' d'equips d'estudiants a empreses, és a dir, sense possibilitat de ser un procés iteratiu.

Com es comenta en aquests articles, la principal virtut d'aquest algorisme és que supera a l'optimitzador de propòsit general IBM CPLEX (2019) en termes de temps: sempre arriba a la solució òptima almenys un 55% més ràpida que CPLEX i arriba a una qualitat del 80% en menys del 20% del temps que cal per construir l'entrada per a CPLEX.

Tal com es comenta a la pàgina web d'ILOG, empresa creadora de l'Optimitzador

CPLEX [5], aquest va rebre el seu nom del mètode Simplex implementat en el llenguatge de programació C, tot i que actualment admet més llenguatges. Va ser desenvolupat originalment per Robert E. Bixby i venut comercialment des de 1988 per CPLEX Optimization Inc. Aquesta va ser adquirida per ILOG el 1997 i posteriorment ILOG va ser adquirida per IBM el gener de 2009 [3]. CPLEX continua essent desenvolupat activament per IBM [4]. Aquest optimitzador resol problemes de programació sencers, problemes de programació lineal molt grans utilitzant l'algorisme Simplex.

L'algorisme Simplex [6] és el mètode clàssic per resoldre el problema d'optimització de la programació lineal. A grans trets consisteix a resoldre sistemes com el que es presenta a la Figura 1, on $\mathbf{c} = (c_1, \ldots, c_n)$ són els coeficients de la funció objectiva ($\mathbf{c} = (2, -1, 3)$ a l'exemple), $\mathbf{c}^T$ el vector transposat dels coeficients, $\mathbf{x} = (x_1, \ldots, x_n)$ són les variables del problema ($\mathbf{x} = (x, y, z)$ al sistema exemple), $\mathbf{A}$ és una matriu $p \times n$ amb els coeficients de les restriccions, que poden ser d'igualtat o de desigualtat; i $\mathbf{b} = (b_1, \ldots, b_p)$ són constants no negatives ($\mathbf{b} = (3, 10, 0, 0)$ al sistema).

$$\begin{array}{ll} \text{maximize} & f(x,y,z) = 2x - y + 3z \\ \text{subject to:} & \begin{cases} x - 2y + z = 3 \\ 3x - y + 4z = 10 \\ y \geq 0, \quad z \geq 0 \end{cases} \end{array}$$

Figura 1: Formula Simplex. Extret de [6].

A més a més, al ser problemes d'optimització també s'ha portat en els últims temps al món del *Reinforcement Learning* (RL, aprenentatge per reforç), un camp molt recent dins la IA i el *machine learning*, com es pot veure en el paper *Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization* [7], on utilitzen aquesta nova tècnica per resoldre dos problemes clàssics, el problema del *Traveling Salesman* amb finestres horàries i el problema d'optimització de *4-Moments Portfolio*.

Com es veu, és un problema força recurrent i hi ha moltes maneres d'afrontar-lo. El principal pas endavant que vol fer aquest projecte és que una vegada obtinguda una solució inicial, poder modificar-la a través de la interacció entre els diferents agents implicats.

1.3 Abast i obstacles

1.3.1 Objectius i subobjectius

Com s'ha comentat a la Secció 1.1.2, el principal objectiu és crear una eina que faciliti la feina d'escollir unes pràctiques en empresa per a estudiants. Per a dur a terme aquest objectiu, el treball s'ha dividit en diferents subobjectius:

Part teòrica:

- Investigar les diferents necessitats que tenen estudiants, professorat i empreses en aquest àmbit.
- Investigar el millor algorisme a utilitzar per generar bones solucions inicials i que sigui fàcilment editable per afegir-li la reassignació.

Part pràctica:

- Implementar l'algorisme que generi bones solucions en un temps raonable.
- Implementar un sistema que deixi triar a l'estudiant entre més d'una opció possible. El sistema haurà d'ajustar la solució tenint en compte què ha triat cada estudiant en una segona assignació.
- Analitzar els resultats obtinguts pel sistema.

1.3.2 Obstacles i riscos

Hi pot haver alguns riscos que impedeixin el correcte funcionament del projecte. Poden aparèixer alguns obstacles durant l'execució del projecte:

- **Data límit del projecte.** Hi ha un termini per lliurar el projecte que s'ha de respectar. Això obliga a prendre decisions dràstiques durant el desenvolupament.

Per tant, haurà d'haver-hi un bon pla i complir els terminis especificats per poder acabar el projecte a temps.

- **Errors en algunes llibreries.** Algunes llibreries utilitzades poden tenir errors en algunes funcions, cosa que faria que el codi fos incorrecte.
- **Inexperiència en el llenguatge de programació o en les llibreries utilitzades.** En cas que es triés utilitzar un llenguatge de programació o unes llibreries que l'investigador no ha utilitzat mai, hauria d'invertir un temps a aprendre aquest nou llenguatge o llibreries. Això pot implicar que el codi no sigui tan clar com hauria de ser.

1.4 Metodologia de treball i eines de seguiment

1.4.1 Metodologia

La metodologia que utilitzaré per al projecte és la metodologia *Kanban*, l'objectiu principal de la qual és gestionar de manera general com es completen les tasques. Kanban és una paraula japonesa formada per dos kanjis, on Kan significa "visual" i Ban correspon a "targeta" resultant la paraula "targetes visuals". Per tant, en aquesta metodologia s'utilitzarà notes visuals, on cadascuna representa una tasca a fer. Les cartes estaran en un tauler amb 3 columnes diferents:

- **To Do.** Compost per totes les tasques que s'han definit, però que encara no s'han iniciat.
- **In Progress.** Compost per totes les tasques que s'estan desenvolupant.
- **Done.** Compost per totes les tasques acabades i provades.

Per controlar el treball, farem servir l'eina que té GitHub [8] per gestionar projectes. És una eina que té tot el necessari per gestionar de manera correcta utilitzant *Kanban* i a més, com utilitzarem GitHub per guardar tots els arxius del treball online i controlar les versions; permet relacionar-ho còmodament amb les diferents targetes utilitzant branques. Aquesta metodologia destaca per ser molt fàcil d'utilitzar i actualitzar, així

com per una tècnica molt visual, que permet veure molt ràpidament l'estat de totes les tasques del projecte.

1.4.2 Seguiment

Com hem comentat a l'anterior punt, utilitzarem un repositori GitHub com a eina de control de versions, que ens permetrà facilitar la disponibilitat del codi (es troba al núvol) i la recuperació de versions anteriors en cas d'errors crítics. Seguint l'ordre que tenim en les targetes comentades a l'anterior apartat, dividirem les branques de la següent manera:

- **Tasques *In Progress*.** Cada tasca que s'estigui desenvolupant es col·locarà en una branca diferenciada. Es tancaran les branques una vegada la tasca es doni per finalitzada i s'hagi provat que funciona correctament.
- ***Master*.** Es trobarà tot el codi desenvolupat i provat.

Finalment, s'utilitzarà la branca *Hotfix* en cas que s'hagin de resoldre un error específic al codi. A més a més, per mantenir un correcte seguiment de la memòria escrita s'utilitzarà Overleaf [11], una eina online per escriure documentació científica utilitzant LaTeX.

Per últim, indicar que es programaran reunions presencials o telemàtiques un cop cada dues setmanes amb el director del projecte. En aquestes reunions es discutirà l'estat del projecte i les tasques a realitzar durant les dues setmanes següents, abans de la següent reunió. En cas que tingui algun problema en el projecte, s'organitzaran reunions extraordinàries.

2. Planificació temporal

Amb l'objectiu de finalitzar aquest treball de final de grau en la data estimada i complir amb els objectius plantejats prèviament, s'ha dissenyat una planificació temporal del projecte.

El treball comença el 14 de setembre de 2020 i la seva finalització està prevista pel dia 25 de gener de 2021. En total, el desenvolupament del projecte es durà a terme al llarg de 132 dies aproximadament i amb una duració estimada de 450 hores. La dedicació diària serà de 4 hores aproximadament, donant un total de 28 hores a la setmana on de dilluns a divendres es duran a terme tasques de desenvolupament del codi o relacionat i els caps de setmana es dedicaran a redactar.

2.1 Descripció de tasques

En aquesta secció es detallen les tasques que es realitzaran durant el transcurs del projecte. Aquestes s'agrupen per blocs per distingir amb major facilitat les diferents fases del projecte. A la Taula 1 es llisten totes les tasques amb la duració, dependències i recursos necessaris, i a la Figura 2 es mostra el diagrama de Gantt. A continuació expliquem tots els blocs i les seves tasques.

2.1.1 Gestió del projecte (GP)

La gestió del projecte és essencial per planificar, definir i documentar el treball a realitzar. També engloba les reunions per la validació i proposta d'objectius setmanals.

S'estima que el grup de gestió del projecte tindrà una duració de 90 hores.

- **GP.1 - Reunions.**
  Per mantenir una correcta comunicació entre investigador i director, es duran a terme reunions d'1 hora cada dues setmanes com a mínim.
- **GP.2 - Abast.**
  Abans de començar amb el projecte és important acotar l'abast del projecte. Per aquest motiu es dedica temps inicial a definir que es vol aconseguir amb el treball, que es vol desenvolupar i quins mitjans són necessaris per fer-ho.
- **GP.3 - Planificació.**
  Per complir els objectius proposats al GP.2, cal realitzar una planificació temporal, així com de recursos i requeriments associats a cada tasca. A més a més també definim els riscs i obstacles i es plantegen tasques alternatives per resoldre'ls.
- **GP.4 - Pressupost.**
  Es realitza un pressupost per quantificar el cost del projecte. Per això es destinaran partides per a cada tasca tenint en compte tots els costos, incloent-hi personal, equip i imprevistos.
- **GP.5 - Informe de sostenibilitat.**
  Es realitzarà un informe de la part de planificació i desenvolupant sobre l'impacte mediambiental, econòmic i social del projecte.
- **GP.6 - Revisió i informe final per GEP.**
  Una vegada completat GP.2, GP.3, GP.4 i GP.5 es revisarà i s'unificarà en un únic document per l'última entrega de GEP.
- **GP.7 - Informe per la reunió de seguiment.**
  El dia 17 de desembre com a màxim caldrà fer una reunió amb el tutor per verificar el seguiment del projecte. Per aconseguir una reunió eficaç es redactarà un informe amb tot el fet fins a aquella data.

2.1.2 Treball previ (TP)

En aquest apartat es realitzaran totes les tasques prèvies al desenvolupament del treball. La duració estimada és de 35 hores.

- **TP.1 - Estudi de l'estat de l'art.**
  Per saber l'estat en el qual es troba l'àmbit d'investigació del treball, cal realitzar una recerca sobre quins treballs s'han fet anteriorment.
- **TP.2 - Preparació de l'entorn de treball.**
  Per treballar correctament durant tot el projecte, caldrà instal·lar i preparar tots programes necessaris.

2.1.3 Generació de la base de dades (DB)

Abans de començar a desenvolupar el sistema, caldrà generar i pensar una base de dades, la qual tindrà una duració estimada de 65 hores.

- **DB.1 - Creació de les preguntes per estudiants i empreses.**
  Prèviament a implementar la Base de Dades, es pensaran i escriuran totes les preguntes per a estudiants i empreses.
- **DB.2 - Creació de la base de dades.**
  Es pensarà com emmagatzemar la base de dades i es dissenyarà el format.
- **DB.3 - Implementar script per automatitzar.**
  Es crearà un script per automatitzar la creació d'estudiants i pràctiques.

2.1.4 Desenvolupament de l'algorisme (DA)

En aquest grup de tasques es farà recerca, desenvoluparà i provarà l'algorisme. El temps esperat per aquest grup és de 210 hores.

- **DA.1 - Aprendre sobre diferents algorismes que podem utilitzar.**
  Tasca dedicada a investigar i aprendre sobre els possibles algorismes que podem utilitzar per decidir quin és més correcte d'implementar pel nostre problema.
- **DA.2 - Implementació i test de l'algorisme principal.**
  S'implementarà i testejarà l'algorisme per assignar estudiants a pràctiques en empresa.

- **DA.3 - Implementació i test de l'algorisme de recol·locació.**
  S'implementarà i testejarà l'algorisme per recol·locar els estudiants una vegada han decidit quines pràctiques prefereixen.
- **DA.4 - Implementació d'scripts d'automatització.**
  Una vegada s'han determinat els algorismes, s'implementarà un conjunt d'scripts per executar-los en conjunt automàticament.
- **DA.5 - Anàlisis dels resultats.**
  S'analitzarà el funcionament dels algorismes.

2.1.5 Documentació del projecte (DC)

Durant el transcurs de tot el projecte, caldrà documentar-ho per la correcta execució de la memòria. S'estima que el grup de documentació del projecte tindrà una duració de 50 hores.

- **DC.1 - Escriure documentació.**
  A tot TFG cal desenvolupar una memòria escrita que documenti adequadament totes les fases del treball. La documentació es realitzarà de forma paral·lela a la resta de treball.
- **DC.2 - Revisió de la documentació.**
  Una vegada acabada la part escrita, es revisarà amb el tutor la documentació.
- **DC.3 - Preparació de la presentació.**
  Una vegada finalitzat tota la documentació, és necessari preparar la presentació pel tribunal que avaluarà el TFG. Per realitzar una bona presentació es generarà un suport per la presentació, així com guió i es realitzaran assajos.

2.2 Recursos

A continuació es descriuran tots els recursos necessaris per fer correctament el treball.

2.2.1 Recursos humans

En aquest projecte es troben quatre rols diferents: cap de projecte, investigador, programador i tester.

No obstant això, tenint en compte que aquest treball serà realitzat només per una persona, serà l'autor l'encarregat d'assumir els diferents rols, segons calgui en cada moment. A la Taula 1 s'exposa les assignacions per cada tasca.

- **Cap de projecte.** S'encarrega de planificar el projecte, liderar les reunions amb l'equip i escriure la documentació.
- **Investigador.** S'encarrega d'investigar tot el necessari per a la correcta realització del projecte. Dissenyar les tècniques i realitza les proves amb els usuaris, avalua els resultats obtinguts i contribueix en la documentació.
- **Programador.** És la persona encarregada d'implementar el sistema.
- **Tester.** S'encarrega de realitzar les proves de validació del sistema implementat. Ha de dissenyar les proves, executar-les i presentar un informe amb els errors trobats.

2.2.2 Recursos materials

A continuació s'exposa tots els recursos materials necessaris per a la correcta realització del projecte. A la Taula 1 s'exposa l'assignació de material a cada tasca.

- **Portàtil.** Ordinador portàtil per realitzar pràcticament totes les tasques del projecte.
- **GitHub [8].** Eina per realitzar el control de versions, guardar el codi al núvol i per gestionar el projecte utilitzant la metodologia *Kanban*.
- **JetBains PyCharm Community Edition [9].** Editor de text per facilitat el desenvolupament del codi en Python [10].
- **Overleaf [11].** Per escriure i guardar la memòria escrita.

Id	Tasca	Temps	Dpds.	R.Humans	R.Materials
GP	**Gestió del projecte**	**90h**			
GP.1	Reunions	20h	-	CP,I,P,T	PC,O
GP.2	Abast	25h	TP.1	CP	PC,O
GP.3	Planificació	15h	GP.2	CP	PC,O,GH,GT
GP.4	Pressupost	7h	GP.3	CP	PC,O
GP.5	Informe sostenibilitat	7h	GP.4	CP	PC,O
GP.6	Revisió i informe final GEP	6h	GP.5	CP	PC,O
GP.7	Informe per la reunió de seguiment	10h	-	CP	PC,O
TP	**Treball previ**	**35h**			
TP.1	Estudi de l'estat de l'art	30h	-	I	PC
TP.2	Preparació de l'entorn de treball	5h	-	P	PC,J,GH
DB	**Generació de la Base de Dades**	**65h**			
DB.1	Creació de les preguntes	20h	TP.2	I,P	PC,O
DB.2	Creació de la Base de Dades	10h	DB.1	P	PC,O
DB.3	Implementar script per automatitzar	35h	DB.2	P	PC
DA	**Desenvolupament de l'algorisme**	**210h**			
DA.1	Aprendre sobre diferents algorismes	30h	TP.2	I	PC
DA.2	Impl. i test de l'alg. principal	70h	DA.1,DB.3	I,P,T	PC,J,GH
DA.3	Impl. i test de l'alg. de recol·locació	60h	DA.2	I,P,T	PC,J,GH
DA.4	Impl. d'scripts d'automatització	30h	DA.3	P,T	PC,J,GH
DA.5	Anàlisis dels resultats	20h	DA.4	I,P,T	PC,O
DC	**Documentació projecte**	**50h**			
DC.1	Escriure documentació	-	-	CP,I,P,T	PC,O
DC.2	Revisió documentació	30h	DC.1	CP	PC,O
DC.3	Preparar presentació	20h	DC.2	CP	PC,O
Total		**450h**			

Taula 1: Recull de les tasques de la planificació. Elaboració pròpia. Llegenda: Dpds. Dependències; (CP) Cap de projecte; (I) Investigador; (P) Programador; (T) Tester; (PC) Ordinador; (O) Overleaf; (GH) GitHub; (J) JetBrains PyCharm; (GT) Gantter.

– **Tomsplanner [12].** Web per generar el diagrama de Gantt.

2.3 Gestió del risc

Per tal de controlar al màxim qualsevol imprevist, cal preveure els possibles riscos i obstacles que ens puguem trobar durant el desenvolupament del TFG.

– **Inexperiència en alguna de les tecnologies.** Inexperiència en alguna de les

tecnologies. Hem de tenir en compte que s'aplicaran tecnologies i coneixements nous, per tant cal contemplar que hi hagi endarreriments per aquest tema. Es tenen en compte per totes les tasques de desenvolupament, les quals tenen un risc alt al diagrama de Gantt.

- **Errors en algunes llibreries.** Hem contat temps extra en les tasques de desenvolupament per si de cas cal mirar llibreries noves
- **Planificació temporal errònia.** Ens hem guardat dues setmanes per si de cas la planificació temporal té errors i és necessari més temps.

Per tal de marcar amb més claredat les tasques que tenen més riscos, s'ha especificat al diagrama de Gantt per a cada tasca el nivell de risc (baix, mig, alt) segons els punts anteriors.

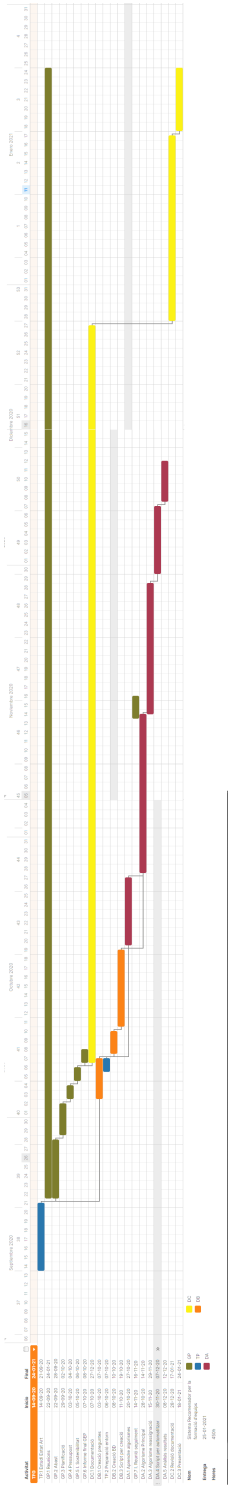

Figura 2: Diagrama de Gantt. Elaboració propia.

3. Gestió Econòmica

3.1 Pressupost

En aquest apartat es defineix el pressupost total del projecte.

3.1.1 Identificació de costos

Per facilitar el càlcul dels costos, els classifiquem en diferents categories:

- **Personal.** Sous del cap del projecte, investigador, programador i tester.
- **Materials.** Material utilitzat pels treballadors.
- **Indirectes i d'energia.** Espai i consum elèctric dels treballadors.

3.1.2 Costos de personal

A partir de la planificació de tasques es calcula el cost de personal, tenint en compte els quatre rols que tenim: Cap de projecte, investigador, programador i tester.

A la Taula 2 es veu els costos per hora de cada rol bruts obtinguts de GlassDoors [13], un portal web que mostra els sous mitjans dels diferents rols que treballen a una zona propera a la de desenvolupament del projecte. Seguidament, també es pot veure el sou una vegada se li ha aplicat la Seguretat Social (per fer-ho fàcil s'ha arrodonit a un augment del 1,3).

Rol	**Cost per hora brut**	**Cost per hora net**
Cap del projecte	25,0 €/h	32,5 €/h
Investigador	20,0 €/h	26,0 €/h
Programador	18,0 €/h	13,5 €/h
Tester	18,0 €/h	23,5 €/h

Taula 2: Costos de personal per cada tasca. Elaboració pròpia.

Posteriorment, a la Taula 3 podem veure el salari total de cada rol, tenint en compte les tasques que fa cadascú.

Tasca	**Cap del projecte**		**Investigador**		**Programador**		**Tester**		**Total Net (€)**
	hores	preu (€)	hores	preu (€)	hores	preu (€)	hores	preu (€)	
GP.1	20	**650,0**	20	**520,0**	20	**470,0**	20	**470,0**	**2110,0**
GP.2	25	**812,5**							**812,5**
GP.3	15	**487,5**							**487,5**
GP.4	7	**227,5**							**227,5**
GP.5	7	**227,5**							**227,5**
GP.6	6	**195,0**							**195,0**
GP.7	10	**325,0**							**325,0**
GP	90	**2925,0**	20	**520,0**	20	**470,0**	20	**470,0**	**4385,0**
TP.1			30	**780,0**	20	**470,0**			**780,0**
TP.2					5	**117,5**			**117,5**
TP	0	**0**	30	**780,0**	5	**117,5**	0	**0**	**897,5**
DB.1			20	**520,0**	20	**470,0**			**990,0**
DB.2					10	**235,0**			**235,0**
DB.3					30	**705,0**			**705,0**
DB	0	**0**	20	**520,0**	60	**1410,0**	0	**0**	**1930,0**
DA.1			30	**780,0**					**780,0**
DA.2			70	**1820,0**	70	**1645,0**	70	**1645,0**	**5110,0**
DA.3			60	**1560,0**	60	**1410,0**	60	**1410,0**	**4380,0**
DA.4					30	**705,0**	30	**705,0**	**1410,0**
DA.5			20	**520,0**	20	**470,0**	20	**470,0**	**940,0**
DA	0	**0**	180	**4680,0**	180	**4230,0**	180	**4230,0**	**12620,0**
DC.1	-	**-**	-	**-**	-	**-**	-	**-**	**-**
DC.2	30	**975,0**							**975,0**
DC.3	20	**650,0**							**650,0**
DC	50	**1625,0**	0	**0**	0	**0**	0	**0**	**1625,0**
Total	140	**4550,0**	250	**6500,0**	260	**6227,5**	200	**4700,0**	**21457,5**

Taula 3: Costos de personal. Elaboració pròpia.

3.1.3 Costos de materials

Durant tota la realització d'aquest treball s'utilitzarà un ordinador portàtil HP [14]. Pel que fa al programari, com que únicament s'utilitzarà software gratuït no cal incloure cap despesa per aquest concepte.

Per calcular l'amortització del portàtil s'utilitzarà la següent fórmula:

$$\frac{cost(e)}{vida\_util(anys) \times 220(dies\_feiners\_any) \times dedicacio/dia(h)} \times duracio\_projecte(h)$$

Aplicant-la al nostre projecte (4 hores diàries i 450 hores de dedicació), obtenim la Taula 4.

Element	**Preu (€)**	**Vida útil**	**Amortització (€)**
Portàtil HP	1000	5 anys	**102,27**

Taula 4: Costos dels materials. Elaboració pròpia.

3.1.4 Costos indirectes i d'energia

En aquest apartat calculem els costos relacionats amb els recursos involucrats amb l'espai de treball.

Com que és un treball realitzat únicament per una persona en espais públics o a casa seva, només cal comptar el cost energètic, resumit a la Taula 5.

Element	**Temps**	**Potència**	**Consum**	**Preu/kWh**	**Preu (€)**
Portàtil HP	450h	6,59W	2,965Wh	0,08	**0,237**

Taula 5: Costos d'energia. Elaboració pròpia.

3.1.5 Contingència

Pel pressupost també hem de tenir en compte possibles imprevistos, per això dedicarem una contingència de 10% per a cada cost, resumit amb la Taula 6.

Tipus de cost	**Cost (€)**	**Contingència (%)**	**Cost final (€)**
Personal	21457,00	10	**23602,70**
Material	102,27		**112,50**
Indirectes i energia	0,24		**0,27**

Taula 6: Contigència. Elaboració pròpia.

3.1.6 Imprevistos

També cal tenir en compte els possibles imprevistos que pot haver-hi, contemplats amb un extra de temps per les diferents tasques.

A més, hi ha la possibilitat que el portàtil necessari per a la realització de tot el treball s'espatlli. Per això es contempla la partida de la Taula 7.

Element	**Preu (€)**	**Probabilitat (%)**	**Cost final (€)**
Portàtil HP	300	10	**30**

Taula 7: Costos dels imprevistos. Elaboració pròpia.

3.1.7 Pressupost final

Una vegada analitzat cada cost, tenim el pressupost final a la Taula 8.

3.2 Control de gestió

Per controlar les desviacions en el pressupost, al finalitzat cada tasca s'actualitzarà en funció de les hores empleades i dels imprevistos. Es calcularan les desviacions a partir

Tipus de cost	**Cost amb contingències (€)**
Personal	23602,70
Material	112,50
Indirectes i energia	0,27
Total	**23715,47**
imprevistos	30,00
Total amb imprevistos	**23745,47**

Taula 8: Pressupost final. Elaboració pròpia.

de les següents fórmules.

- **Desviació en la realització de les tasques.**

$$(Hores\ estimades - Hores\ reals) \times Cost\ real$$

- **Desviació dels costos en recursos humans per tasca.**

$$(Cost\ estimat - Cost\ real) \times Hores\ reals$$

- **Desviació dels costos recursos materials.**

$$Cost\ material\ estimat - Cost\ material\ real$$

- **Desviació dels costos indirectes.**

$$Cost\ indirecte\ estimat - Cost\ indirecte\ real$$

- **Desviació total dels imprevistos.**

$$Cost\ imprevist\ estimat - Cost\ imprevist\ real$$

- **Desviació total en hores.**

$$Hores\ estimades - Hores\ reals$$

- **Desviació total dels costos.**

$$Cost\ total - Cost\ total\ real$$

4. Informe de sostenibilitat

Per realitzar un correcte TFG també cal tenir en compte tot el que veure amb la sostenibilitat del projecte. Per saber l'impacte que tindrà aquest projecte en l'ecosistema en el qual vivim s'ha realitzat un informe sobre la dimensió econòmica, mediambiental i social.

4.1 Dimensió econòmica

El cost estimat d'aquest projecte únicament inclou els recursos necessaris per aconseguir els objectius marcats en el termini de temps necessari, per tant no hi ha malbaratament de diners ni recursos.

A més a més, és un projecte que comporta un gran benefici pel sistema acadèmic, per què permet substituir una feina molt feixuga que es faria o amb paper o amb ordinadors, cosa que implica a la llarga un malbaratament de recursos elevat.

Per últim, cal destacar que el sistema final serà codi obert i reutilitzable, per tant qualsevol que vulgui podrà utilitzar-lo pel seu benefici, cosa que permet estalviar costos de desenvolupament en futurs projectes semblants.

4.2 Dimensió mediambiental

Al ser un projecte de software, no tindrà cap impacte mediambiental a part de l'electricitat necessària per desenvolupar-ho i executar-ho. Cal tenir en compte que és un sistema

que està pensat perquè s'utilitzi principalment en centres educatius, per tant el cost d'executar-ho serà mínim perquè ja tenen servidors funcionant tot el temps per altres coses.

A més, com s'ha comentat a l'anterior apartat, permetrà reduir el consum d'electricitat o paper, amb tot l'impacte mediambiental que això implica.

4.3 Dimensió social

Personalment, d'ençà que vaig començar l'especialitat de Computació m'ha interessat molt com algorismes poden millorar la vida de les persones i en definitiva de la societat.

Amb aquest projecte l'objectiu principal és ajudar als centres docents, estudiants i empreses a fer-los la vida més fàcil, per tant, tal com s'ha comentat en els anteriors apartats, té un clar component social i es vol que tingui un impacte positiu en els centres acadèmics.

Per últim, el fet que sigui el codi sigui públic, també permet a futurs projectes enriquir-s'hi.

5. Preliminars

Abans de començar explicant el desenvolupament del projecte, és necessari introduir un conjunt de conceptes bàsics per entendre'l.

Per això, en aquest apartat es donen un conjunt de definicions i els coneixements essencials de les tècniques, algorismes i recursos utilitzats durant el transcurs d'aquest.

5.1 Algorismes descartats

L'algorisme és la part central del projecte i on s'ha dedicat més temps. Abans de començar la fase d'implementació, l'autor i el tutor van contemplar diferents algorismes que es podien utilitzar per resoldre el problema.

El problema d'assignació d'estudiants és un problema NP-Complet el qual es pot afrontar des de diferents camps. Principalment es va contemplar entendre'l com un problema d'optimització o encarar-ho cap al camp del Reinforcement Learning.

Tractant el problema des del punt de vista de l'optimització combinatòria, utilitzant un algorisme heurístic per resoldre el problema era l'opció sobre el paper més fàcil, ja que teníem disponible un projecte similar [2] i ens permetia centrar-nos en la reassignació d'estudiants que és el principal objectiu del projecte.

Així i tot, intentant buscar solucions més innovadores finalment es va decidir encarar el projecte cap al camp del Reinforcement Learning.

5.2 Reinforcement Learning

El Reinfocerment Learning (RL) o aprenentatge per reforç [7] és un camp del machine learning centrat en la forma en què els agents intel·ligents han d'actuar en un entorn per maximitzar la noció de recompensa acumulativa. El RL és un dels tres paradigmes bàsics d'aprenentatge automàtic, juntament amb l'aprenentatge supervisat i l'aprenentatge no supervisat.

Per formar un entorn de RL es necessari de dos components, l'agent i l'entorn, com podem veure a la Figura 3.

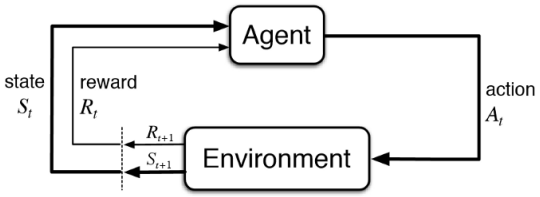

Figura 3: Esquema de funcionament del RL. Extret de https://bit.ly/39vcBVR.

L'entorn és l'espai on l'agent actua i pren decisions, l'algorisme de RL es troba a l'agent i a l'entorn es troben totes les regles del joc (específic per a cada problema).

A continuació teniu algunes definicions d'interès:

- **Acció (A).** Definit per l'entorn, tot el conjunt d'accions que pot fer l'agent.
- **Estat (S).** Situació actual retornada per l'entorn.
- **Reward (R).** Retorn intermedi retronat per l'entorn per avaluar l'última acció presa. Aquest per tenir valor positiu o negatiu (penalització).
- **Política (π).** Estratègia que l'agent utilitza per determinar quina acció és la més correcta segons l'estat on estigui.
- **Valor (V).** El valor que s'espera a llarg termini a diferència del Reward que és a curt termini. $V_\pi(s)$ es refereix al retorn a llarg termini de l'estat s sota la política π.

– **Q-Valor o valor-acció (Q).** Semblant a Valor, però té en compte un paràmetre més, l'acció actual. *Qπ(s,a)* es refereix al retorn a llarg termini de l'estat s trien l'acció a sota la política π.

El RL és la ciència de fer decisions òptimes utilitzant l'experiència prèvia, un entrenament. Aquest procés té els següents passos:

1. Observació de l'entorn.
2. Decidir com actuar utilitzant una estratègia.
3. Actuar.
4. Rebre un reward o una penalització.
5. Aprendre de l'experiència i redefinir l'estratègia.
6. Iterar fins a obtenir l'estratègia òptima.

Com es pot intuir, hi ha una gran varietat d'algorismes de RL, a continuació explicarem resumidament el funcionament dels dos que hem utilitzat en el nostre projecte.

5.2.1 Q Learning

El primer algorisme que vam provar pel projecte va ser el clàssic Q Learning. És un algorisme off-policy (aprèn basant-se en l'acció a^* obtinguda d'una altra política) i model-free (es basa en prova i error per actualitzar els seus coneixement) basat en l'equació de Bellman de la Figura 4, on $\mathbb{E}$ expressa l'expectativa i λ expressa el descompte.

$$v(s) = \mathbb{E}[R_{t+1} + \lambda v(S_{t+1}) | S_t = s]$$

Figura 4: Equació de Bellman. Extret de https://bit.ly/3skcwwR.

Aquesta funció es pot reescriure segons la forma de Q-Valor de la Figura 5 on el valor òptim de Q-Valor, Q^* es pot expressar com indica la Figura 6

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \ldots | s,a]$$
$$= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s',a') | s,a]$$

Figura 5: Equació de Bellman segons Q-Valor. Extret de https://bit.ly/3qezLpW.

$$Q^{*}(s,a) = \mathbb{E}_{s'}[r + \lambda \max_{a'} Q^{*}(s',a') | s,a]$$

Figura 6: Valor òptim de Q-Valor. Extret de https://bit.ly/3qezLpW.

L'objectiu de Q-Learning és maximitzar el Q valor. Es fa mitjançant dos processos, la iteració de política i la iteració del valor.

La iteració de política és el bucle entre l'avaluació i la millora de la política, que són dos càlculs diferenciats. Primer, l'avaluació de la política estima la funció de valor V amb una política greedy obtinguda de l'última millora de política. En canvi, la millora de política s'obté maximitzant V per cada estat i està basat en l'equació de Bellman que hem vist abans. Aquesta iteració es farà fins que els dos valors convergeixen (veure Figura 7).

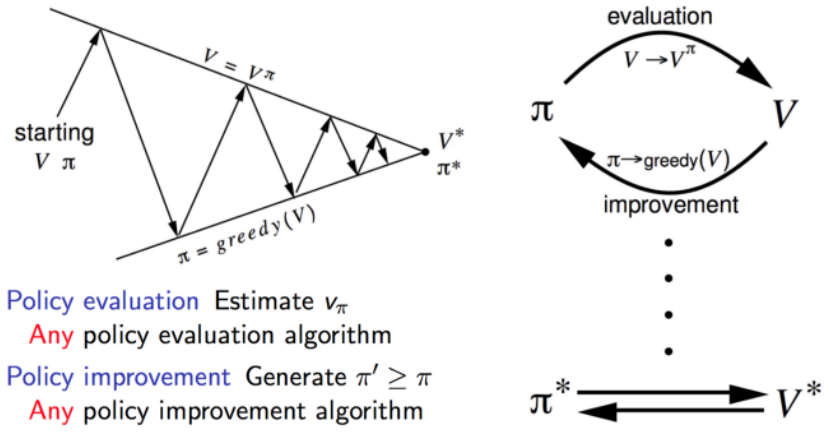

Figura 7: Iteració de política.

Extret de https://bit.ly/39tD4mB.

La iteració de valor actualitza V està basada en l'equació optima de Bellman de la Figura 8.

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_*(s')\right] \end{aligned}$$

Figura 8: Equació Bellman óptima. Extret de https://bit.ly/35AlT1B.

Una vegada la iteració ha convergit, la política òptima es deriva directament aplicant una funció argument-max per a tots els estats.

Com es pot intuïr, aquests dos mètodes necessiten conèixer la probabilitat de transició p, cosa que convertiria Q-Learning en un algorisme model-based.

Com que no ho és, pateix d'un problema d'escalabilitat que és solucionat canviant la funció d'actualització per la funció de la Figura 9, on α expressa la taxa d'aprenentatge. Això ens permet obviar p.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Figura 9: Equació d'actualització de Q Learning. Extret de https://bit.ly/2Low5U9.

A la Figura 10 podem veure el pseudocodi de Q Learning.

5.2.2 Algorisme A2C

L'A2C [21] o Advantage Actor Crític és una versió alternativa a la implementació asíncrona d'A3C. A2C és una implementació síncrona i determinista que espera que cada actor acabi el seu segment d'experiència abans d'actualitzar-la, fent una mitjana de tots els actors. Això fa servir GPU de manera més eficaç a causa de les mides de lots més grans.

L'A2C és el que es coneix com un model Deep RL, una màquina de mapatge d'entrada-sortida igual que qualsevol altre model de classificació NN o regressió [22]. En comptes

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:
Sates $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Start in state $s \in \mathcal{X}$
    while s is not terminal **do**
      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow T(s, a)$ ▷ Receive the new state
      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
      $s \leftarrow s'$
  return Q

Figura 10: Pseudo-codi de Q Learning. Extret de https://bit.ly/3qkhwzD.

de mapejar imatges o text a categories, un model Deep RL mapeja estats a accions o estats a valors-estat. Un A2C fa les dues coses com es pot veure a la Figura 11.

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)] Q_w(s_t, a_t) \\ &= \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q_w(s_t, a_t)]\end{aligned}$$

Figura 11: Funció d'actualització que utilitza A2C. Extret de https://bit.ly/3snlB83.

El valor Q es pot aprendre parametritzant la funció Q amb una xarxa neuronal (denotada pel subíndex w anterior).

Això ens porta a parlar del funcionament d'aquest mètode, el qual té dues funcions diferents:

– **El crític.** Una estimació de quin reward s'espera obtenir a partir d'aquest moment, el valor del estat. Aquest podria ser el valor-acció (Q valor) o el valor d'estat V.

– **L'actor.** Que actualitza la distribució de les polítiques en la direcció suggerida pel crític (com ara amb gradients de polítiques).

Aquestes funcions es parametritzen amb xarxes neuronals. A la derivació anterior, la xarxa neuronal del crític parametritza el valor Q, de manera que s'anomena Q actor crític, el seu pseudocodi es pot veure a la Figura 12.

Algorithm 1 Q Actor Critic

Initialize parameters s, θ, w and learning rates α_θ, α_w; sample $a \sim \pi_\theta(a|s)$.
for $t = 1 \dots T$: **do**
  Sample reward $r_t \sim R(s, a)$ and next state $s' \sim P(s'|s, a)$
  Then sample the next action $a' \sim \pi_\theta(a'|s')$
  Update the policy parameters: $\theta \leftarrow \theta + \alpha_\theta Q_w(s, a) \nabla_\theta \log \pi_\theta(a|s)$; Compute the correction (TD error) for action-value at time t:
    $\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$
  and use it to update the parameters of Q function:
    $w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$
  Move to a $\leftarrow a'$ and s $\leftarrow s'$
end for

Figura 12: Pseudocodi del Q actor crític. Extret de https://bit.ly/3qkhZln.

Per tot això, quan l'agent arriba a un estat generarà dues imatges (l'actriu i la crítica). Com tot mètode RL, donat un estat i una acció, l'entorn li retornarà un reward. Aquest conjunt d'estat-acció-reward genera una única observació que és guardada una vegada l'agent ja està al següent estat. Per tant, el reward s'associa amb l'estat i l'acció que el precedia.

Així mateix, l'agent torna a repetir el procés anterior dues vegades més fins que té un conjunt de tres observacions, on les analitzarà en conjunt.

Abans que pugui calcular el seu crític interior, ha de calcular quants punts realment rebria de cada estat donat, però no ho pot saber realment, ja que ho fa abans de finalitzar tot el procés (en el nostre cas seria quan ha assignat tots els estudiants).

En l'A2C, a diferència d'altres models com Monte Carlo, calcula una estimació de quants punts aconseguiria mitjançant els crítics que s'ha anat generant, d'aquesta manera pot recalcular els valors "reals" dels estats anteriors.

Monte Carlo estima les "etiquetes" objectiu durant el rollout d'una trajectòria i sumant

les recompenses de cada estat en endavant. En canvi, A2C trunca aquesta trajectòria i la substitueix per una estimació del seu crític, reduint la variància d'estimació i li permet funcionar contínuament, tot i que a costa d'introduir una mica de biaix.

D'aquesta manera, l'algorisme revisa les observacions que ha anat obtenint per aprendre de les diferències cada tres passos en comptes d'esperar al final. Aquest conjunt de tres observacions és un minibatch minúscul i autocorrelat de dades de formació etiquetades.

Per reduir aquesta autocorrelació, molts A2C executen múltiples agents en paral·lel, apilant les seves experiències abans de pujar-los a una xarxa neuronal compartida.

Una vegada sintonitza el seu crític interior, A2C ajusta l'actor interior.

Un simple algorisme de gradient de política analitzaria els rendiments reals després d'una acció i ajustaria la seva política per fer més bons els rendiments.

En canvi, en lloc d'ajustar la seva política en resposta als rendiments totals que va obtenir fent una única acció, ajusta les accions als rendiments relatius de fer aquesta acció. Això s'anomena avantatge, la seva equació la tenim a la Figura 13.

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

Figura 13: Equació d'avantatge. Extret de https://bit.ly/2LPZB4I.

L'A2C l'utilitza per fer més probable que escolli accions trobant resultats sorprenentment bons. A més també l'aprofita com a error, on utilitza la mateixa quantitat per empènyer la crítica interna per fer millors estimacions dels valors d'estat.

Ara podem mostrar com es calcula la pèrdua total, aquesta és la funció que minimitzem per millorar el nostre model:

$$perdua\_total = perdua\_accions + perdua\_valor - entropia.$$

On l'entropia fa referència a la propagació de la distribució de l'acció i serveix per fomentar encara més l'exploració.

Cal tenir en compte que estem empenyent gradients de tres tipus qualitativament diferents a través d'un sol NN, cosa que és eficient, però pot dificultar la convergència.

A la Figura 14 es pot veure el pseudocodi de l'algorisme A2C.

Algorithm 1 Advantage actor-critic - pseudocode

// Assume parameter vectors θ *and* θ_v
Initialize step counter $t \leftarrow 1$
Initialize episode counter $E \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta)$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta_v) & \text{for non-terminal } s_t \text{ //Bootstrap from last state} \end{cases}$
  for $i \in \{t-1,\ldots,t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ: $d\theta \leftarrow d\theta + \nabla_\theta \log \pi(a_i|s_i;\theta)(R - V(s_i;\theta_v)) + \beta_e \partial H(\pi(a_i|s_i;\theta))/\partial\theta$
    Accumulate gradients wrt θ_v: $d\theta_v \leftarrow d\theta_v + \beta_v(R - V(s_i;\theta_v))(\partial V(s_i;\theta_v)/\partial\theta_v)$
  end for
  Perform update of θ using $d\theta$ and of θ_v using $d\theta_v$.
  $E \leftarrow E+1$
until $E > E_{max}$

Figura 14: Pseudocodi d'A2C. Extret de https://bit.ly/3oMJCD6.

5.3 Eines

Per programar l'algorisme de Reinforcement Learning així com tots els scripts complementaris s'han necessitat un conjunt d'eines que explicarem a continuació.

5.3.1 Python

El llenguatge utilitzat per desenvolupar tot el projecte ha sigut Python [10], un llenguatge de programació interpretat, d'alt nivell i per una gran varietat d'usos. És un llenguatge molt ràpid d'entendre i escriure cosa que el fa perfecte pel camp de l'IA i el deep learning.

A més a més, és un llenguatge que potencia molt el codi obert mitjançant l'ús de llibreries. Això ha permès que augmenti molt la seva comunitat fins a convertir-se amb uns

dels llenguatges més utilitzats avui en dia [16] i amb una major quantitat i qualitat de llibreries de tot tipus que simplifiquen molts càlculs, inclosos tots els que tenen a veure amb IA i deep learning.

Per aquest projecte s'han utilitzat vàries llibreries, exposades a la Secció 5.3.2.

5.3.2 Llibraries

Les llibreries utilitzades durant el transcurs del projecte són les següents.

OpenAI Gym

Gym [17] és una llibreria creada per OpenAI amb tot un conjunt d'eines per desenvolupar i comparar algorismes de Reinforcement Learning.

La principal virtut que té i la principal raó per la qual l'hem utilitzat per al projecte és per crear un entorn personalitzat, específic per al nostre domini problemàtic.

Gym ens proporciona la següent interfície per crear el nostre entorn:

```
import gym
from gym import spaces

class CustomEnv(gym.Env):
  """Custom Environment that follows gym interface"""
  metadata = {'render.modes': ['human']}

  def __init__(self, arg1, arg2, ...):
    super(CustomEnv, self).__init__()
    # Define action and observation space
    # They must be gym.spaces objects
    # Example when using discrete actions:
    self.action_space = spaces.Discrete(N_DISCRETE_ACTIONS)
    # Example for using image as input:
    self.observation_space = spaces.Box(low=0, high=255,
```

```
        shape= (HEIGHT, WIDTH, N_CHANNELS), dtype=np.uint8)

  def step(self, action):
    # Execute one time step within the environment
    ...
  def reset(self):
    # Reset the state of the environment to an initial state
    ...
  def render(self, mode='human', close=False):
    # Render the environment to the screen
    ...
```

Llista 5.1: Codi base d'un entorn de Gym. Extret de https://bit.ly/39zZfYa

Amb les quatre funcions bàsiques:

- **Constructor**. On definirem els valors bàsics d'un entorn, com l'`action_space`, que és el conjunt d'accions possibles que té l'entorn i l'`observation_space`, que és el conjunt d'estats possibles que té l'entorn.
- **Step**. Donada una acció com a paràmetre retornarà l'estat següent, el reward, si és final i una observació (pot retornar-la buit).
- **Reset**. Reinicialització de l'entorn a l'estat inicial.
- **Render**. Per visualitzar l'estat actual.

L'`action_space` i l'`observation_space` són de classe `gym.spaces`, una classe especialment dissenyada per facilitar els càlculs amb aquests dos objectes. Aquesta classe té quatre tipus possibles:

- `gym.spaces.Discrete`. Genera un espai amb n punts discrets, cadascun dels quals amb un valor a l'interval $[0, n-1]$.
- `gym.spaces.MultiDiscrete`. Genera un espai de k dimensions amb diferent nombre de punts discrets.
- `gym.spaces.Box`. Genera un espai multidimensional limitat.

– `gym.spaces.Tuple`. Genera un espai amb forma de tupla.

Gym ens ha permès construir més fàcilment l'entorn específic pel problema, el qual explicarem a la Secció 6.3.

Stable Baselines

Stable Baselines [18] és una llibreria amb un conjunt d'implementacions millorades d'algorismes de Reinfocement Learning basats en OpenAI Baselines [19].

Com s'explica a l'article escrit pels autors *Stable Baselines: a Fork of OpenAIBaselines — Reinforcement Learning Made Easy* [20] aquesta llibreria la van crear perquè a l'OpenAI Baselines faltaven molts comentaris, noms significatius de variables, consistència (sense estil comú entre tot el codi) i tenia molts codis duplicats.

Aquesta llibreria inclou algorismes de RL tals com A2C, PPO, TRPO, DQN, ACKTR, ACER i DDPG. A la Taula 9 es detallen les seves principals característiques.

Nom	**(1)**	**Recurrent**	***Box*(2)**	***Discrete*(2)**	***MultiDiscrete*(2)**	***MultiBinary*(2)**	**(3)**
A2C	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ACER	Yes	Yes	No	Yes	No	No	Yes
ACKTR	Yes	Yes	Yes	Yes	No	No	Yes
DDPG	Yes	No	Yes	No	No	No	Yes
SQN	Yes	No	No	Yes	No	No	No
GAIL	Yes	No	Yes	Yes	No	No	Yes
HER	Yes	No	Yes	Yes	No	Yes	No
PPO1	Yes	No	Yes	Yes	Yes	Yes	Yes
PPO2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SAC	Yes	No	Yes	No	No	No	No
TD3	Yes	No	Yes	No	No	No	No
TRPO	Yes	No	Yes	Yes	Yes	Yes	Yes

Taula 9: Algorismes implementats a Stable Baselines. Extret de [18]. Llegenda: (1) Refactoritzat; (2) És compatible amb; (3) MultiProcesament.

A més a més també inclou wrappers útils, per exemple per al preprocessament o el multiprocessament.

Pel nostre projecte hem utilitzat l'A2C perquè és l'algorisme que té compatibilitat amb tot, expliquem el seu funcionament a la Secció 5.2.2.

Numpy

Numpy [23] és una de les llibreries més populars de Python, ja que té una gran quantitat d'eines de computació científica.

Numpy permet l'ús de matrius de gran mida i multidimensionals així com una gran col·lecció de funcions matemàtiques d'alt nivell per operar-les.

Com s'explicarà a la Secció 6.3.3, les matrius ens han permès representar l'estat com a imatge.

Stadistics

Stadistics [24] és una llibreria bàsica de Python amb implementació de funcions i algorismes d'estadística.

Aquesta llibreria l'hem utilitzat per generar l'estadística dels results de l'algorisme. Com no hem necessitat càlculs estadístics molt complexos ens hem decantat per utilitzar la llibreria més bàsica que té Python.

Pandas

Pandas [25] és una llibreria d'anàlisi i manipulació de dades que hem utilitzat per exportar i importar dades entre els nostres scripts utilitzant arxius CSV.

6. Implementació final

En aquest capítol es mostra la implementació final del projecte. Per fer més fàcil la lectura i el seguiment s'han obviat alguns temes. Tot el coneixement previ per entendre tota la implementació s'explica al capítol anterior de preliminars. Teniu accés al codi complet al Github del projecte, https://bit.ly/2XNn9tN.

6.1 Qüestionari a estudiants i empreses

Una part fonamental del projecte ha sigut dissenyar el conjunt de preguntes que se li fan a estudiants i empreses per determinar l'assignació.

En el nostre cas i al ser l'autor i el tutor del camp de la informàtica, s'han orientat al sector IT, però el sistema és fàcilment modificable per afegir-li tantes més preguntes com es vulgui.

Les preguntes i el format de les respostes que han d'omplir els estudiants són les que trobem a la Taula 10.

Les preguntes i el format de les respostes que han d'omplir els estudiants són les que trobem a la Taula 11.

Com es pot veure, les preguntes tant per l'estudiant com l'empresa s'agrupen en dos tipus, descriptives i de preferència, les quals ens permeten obtenir les puntuacions de compatibilitat, estretament relacionat amb el reward del nostre problema. Expliquem tot el càlcul a la Secció 6.3.4.

Pregunta	**Format de la resposta**
Descriuen l'estudiant	
Nom	String
Edat	Int
Any d'universitat	Float
Nota mitjana	Float
Universitat	Text
Habilitats	Conjunt d'String
Llengües que parla	Conjunt d'String
Experiència laboral	Conjunt d'String
Experiència voluntariat	Conjunt d'String
Descripció	String
Preferències	
Localització	Coordenades
Remot o presencial	Bool, True si Remot, sinó False
Salari Mínim	Int
Tipus de pràctiques	Conjunt d'String

Taula 10: Preguntes a estudiants. Elaboració pròpia.

Per poder provar diferents casos fàcilment s'ha creat un script que genera les respostes ràpidament, expliquem la seva implementació a la Secció 6.5.1.

6.2 Definició del problema

Abans de començar a implementar, és necessari definir el problema més específicament per orientar-lo cap al RL. Això implica posar les normes bàsiques que tindrà el nostre projecte, els trets fonamentals de les quals expliquem en el següent apartat.

6.2.1 Assignació i reassignació

El nostre problema es pot separar en dues parts diferenciades, cadascuna amb característiques particulars:

Pregunta	**Format de la resposta**
Descriuen el projecte	
Nom projecte	String
Nom empresa	String
Número participants màxim	Int
Localització	Coordenades
Remot o presencial	Bool
Tipus	String
Salari	Int
Descripció	String
Preferencies	
Edat	Int
Any d'universitat	Float
Nota mitjana	Float
Universitat	Text
Habilitats	Conjunt d'String
Llengues que parla	Conjunt d'String
Experiencia laboral	Conjunt d'String
Experiencia voluntariat	Conjunt d'String

Taula 11: Preguntes a empresa. Elaboració pròpia.

- **Assignació.** Primera part, on s'assignaran els estudiants per primera vegada i se'ls hi proporcionarà n opcions, tantes com l'usuari vulgui.
- **Reassignació.** Segona part, una vegada els estudiants han definit l'ordre de preferència de les opcions de la primera assignació. En aquesta part el sistema assignarà per última vegada els estudiants amb una única opció final. Aquesta reassignació la farà en ordre de nota mitjana, l'estudiant amb millor mitjana serà el primer a ser assignat, després el segon i així successivament.

Després de contemplar diferents alternatives, finalment es va triar què el mateix sistema fes les dues parts ja que tenen moltes semblances. Els canvis particulars per permetre aquesta doble funcionalitat els exposem a la Secció 6.4.4.

6.2.2 Normes per assignar estudiants

Com s'ha comentat anteriorment, una de les claus del nostre projecte i la qual el diferencia de projectes anteriors és la idea de facilitar a l'estudiant n opcions alternatives perquè pugui elegir.

Aquesta assignació es podria encarar de dues maneres:

- **En una única execució.** El sistema generarà en una única execució les n opcions possibles per cada estudiant.
- **En n execucions.** El sistema s'executarà n vegades, definint en cadascuna una opció per tots els estudiants.

Al ser un problema relativament senzill, vam decidir tirar per la primera de les opcions, ja que simplificava l'entrenament de l'agent.

Seguidament calia posar les normes bàsiques per saber quan es podia assignar o no un estudiant a un projecte, les quals resumin a continuació:

- **El projecte no pot superar les places màximes.** Com és natural, cada projecte té un nombre de places màxim, el qual no es pot superar en cap de les opcions.
- **El projecte no pot tenir assignat el mateix estudiant en la mateixa opció.** Totes les places d'una mateixa opció d'un projecte han d'estar assignades a estudiants diferents.
- **Cada opció d'un estudiant és de diferents projectes.** Cada opció alternativa que se li presenti a cada estudiant seran diferents projectes, per tant no podem tenir el mateix projecte en diverses opcions del mateix estudiant.

6.2.3 Estat, estat inicial i accions possibles

Com implementarem un algorisme de RL cal també definir l'estat, l'estat inicial i accions possibles. L'estat és fàcil de definir, ja que ens cal definir una estructura que representi cada opció de cada estudiant i quin projecte ha sigut assignat a cadascuna.

L'estat inicial i les accions possibles, sí que tenim diferents alternatives:

- **Estat inicial com conjunt buit.** Cap projecte ha estat assignat inicialment a cap estudiant. Per aquesta opció es van contemplar les següents acció per canviar d'estat:
  - **Assignar un estudiant a un projecte.** A una opció en concret o a qualsevol.
- **Estat inicial com un conjunt totalment assignat**. On totes les opcions dels estudiants tenen un projecte assignat de manera més o menys aleatòria. Per aquest cas es va contemplar les següents accions per canviar d'estat:
  - **Assignar o treure un estudiant a un projecte.** A una opció en concret o a qualsevol.
  - **Fer intercanvi d'estudiants entre projectes.**

Amb la idea de facilitar al màxim la programació es va tirar per l'opció d'estat inicial buit amb una única acció d'assignar un estudiant a un projecte. Aquesta opció al ser incremental fa més fàcil controlar les restriccions, cosa que simplica la programació. La segona alternativa no és així, ja que calia comprovar les restriccions de molts grups de parelles.

Finalment, calia decidir entre aquests dos formats d'acció:

- **Assignar un estudiant a un projecte a una opció en concret.**
- **Assignar un estudiant a un projecte a qualsevol opció.**

Amb la intenció de fer l'espai d'accions més petit es va elegir la segona opció, per tant seria l'entorn el que hauria de decidir on assigna l'estudiant que demana l'acció.

6.3 Entorn

L'entorn és l'espai on l'agent actua i pren decisions i és on es troben totes les regles del problema. Gym ens facilita la implementació, però així i tot hem hagut de programar

les funcions fonamentals que donen cos al problema i evita que l'agent se surti de l'espai del problema.

6.3.1 Constructor, espai d'accions i espai observable

El constructor és la part on es defineixen tot un conjunt de característiques del problema. En el nostre cas s'han definit les següents estructures:

- `self.students`.
  Un vector de mida *numero_estudiants* amb el conjunt de respostes dels estudiants. Aquest conjunt és importat mitjançant un JSON.
- `self.projects`.
  Un vector de mida *numero_projectes* amb el conjunt de respostes de les empreses. Aquest conjunt és importat mitjançant un JSON.
- `self.action_space`.
  Defineix l'espai d'accions. Com vam comentar a la Secció 5.3.2, l'`action_space` ens determina quantes accions té el nostre problema. En el nostre cas en tenir només un tipus d'acció (assignar un estudiant a un projecte) genera un espai d'accions que té tantes accions com combinacions estudiant-projecte, per tant la dimensió de l'espai d'accions és $numero\_estudiants^{numero\_projectes}$. L'espai queda definit per un MultiDiscrete de Gym de la següent manera:
  `spaces.MultiDiscrete((len(students), len(projects)))`.
- `self.withImage`.
  És un valor booleà que determina si l'estat es representa i envia a l'agent en forma d'imatge (`withImage=1`) o no (`withImage=0`). Tot això ho veurem extensament a la Secció 6.3.3. Si és amb forma d'imatge, el constructor crearà una estructura extra:
  - `self.stateImageSize`.
    Mida de la imatge que generarà.
- `self.mode`.
  És un valor booleà que ens determina de quina manera s'executa l'entorn. N'hi ha dues possibilitats:

- **Mode 0**. Mode on assigna els estudiants per primera vegada.
- **Mode 1**. Mode on reassigna els estudiants tenint en compte les preferències d'assignació que han seleccionat anteriorment.

A més aquest mode també inicialitzarà una estructura més:

- `self.studentsAlreadyAssigned`.
  És un vector amb els IDs d'estudiants que ja s'han pogut assignar prèviament. Això es fa per simplificar i retallar molt el temps d'execució de la reassignació. Ho explicarem detalladament a la Secció 6.4.4.

– `self.numberOptions`.
Nombre d'opcions que tindrà cada estudiant per elegir una vegada acabi tot el procés d'assignació. Si és mode 1 tindrà només una opció, ja que és una assignació final.

– `self.observation_space`.
Definició de l'espai observable. Com hem comentat anteriorment, l'espai observable és tot el conjunt d'estats on pot arribar l'agent el qual és diferents segons si utilitza imatge o no:

- **Sense imatge**. Es calcularà segons la quantitat de projectes, estudiants i opcions que es vol generar, amb una mida de

$$(numero\_projectes + 1)^{numero\_estudiants \times numero\_opcions}$$

. L'espai queda definit per un MultiDiscrete de Gym de la següent manera: `spaces.MultiDiscrete([len(projects) + 1] * (len(students) * numberOptions))`.

- **Amb imatge**. Amb imatge l'espai canvia lleugerament perquè passa a ser segons la mida de la imatge que es passa. L'espai queda definit per un Box de Gym de la següent manera:

```
spaces.Box( low=0,
high=255,
shape=(self.stateImageSize,
    self.stateImageSize, 3),
    dtype=np.uint8)}.
```

Llista 6.2: Definició de l'espai observable quan s'utilitza imatge.

- `self.state`.
  Variable que guarda l'estat on està el sistema. S'inicialitza en forma d'una matriu de 2 dimensions de mida `len(self.estudiants)` × `self.numberOptions` amb tots els valors a -1 (no s'hi ha assignat cap projecte a l'opció de l'estudiant). Cada casella indica quin projecte té assignat cada estudiant en cada opció.

- `self.assigned`.
  Un vector de mida `self.numberOptions` que indica quans estudiants han estat assignats a cada opció.

- `self.studentsAssignedToProject`.
  Es tracta d'una matriu de 3 dimensions que indica quants estudiants estan assignats a cada projecte i a cada opció. La seva mida és `len(self.projects)` × `self.numberOptions` × *numero_estudiants_assignats_a_projecte*.

- `self.inSameState`.
  Valor sencer inicialitzat a 0 que indica quantes vegades seguides porta el sistema al mateix estat. Aquesta dada és interessant per determinar quan un estat és final. Ho veurem a la Secció 6.3.5.

Per últim, i si el mode = 1, el constructor mitjançant la funció `self._assignStudents()` assignarà els estudiants que ja es troben assignats a `self.studentsAlreadyAssigned` als seus respectius projectes, modificant les variables `self.state`, `self.assigned` i `self.studentsAssignedToProject`.

6.3.2 Canvi d'estat

L'entorn també és l'encarregat d'executar el canvi d'estat segons l'acció que proposa l'agent i per tant de comprovar les restriccions de la Secció 6.2.2 que hem definit. Dins l'entorn serà la funció `step(self, action)` passant-li l'acció com a paràmetre qui ho farà.

Com que a la Secció 6.2.3 vam decidir que l'acció no defineix a quina opció d'aquell estudiant assigna aquest projecte, l'entorn comprovarà per cadascuna de les opcions si en alguna es compleix. D'aquesta manera intentarà primer assignar l'estudiant a aquell projecte per l'opció 0, si no per la 1 i així fins a arribar a l'última opció de l'estudiant.

Per cada opció es comprovarà el següent:

- **Comprova si té assignat o no un projecte per aquella opció.**
  Mirant `self.state`.
- **Comprova si el projecte és ple per aquella opció.**
  Mirant `self.studentsAssignedToProject`.
- **Comprova si el projecte ja ha sigut assignat al mateix estudiant en alguna altra opció.**
  Mirant `self.state`.

Si totes les condicions se satisfan, s'assignarà l'estudiant a aquell projecte, modificant `self.state`, `self.studentsAssignedToProject` i `self.assigned` per mantenir la cohenrència de les dades. S'assignarà també el valor de reward correcte mitjançant la funció `self._rewardCalculation(studentNumber, projectNumber, option)` (comentarem aquesta funció a la Secció 6.3.4). Es comprovarà si l'estat següent és final mitjançant `self._isDone()` (comentarem aquesta funció a la Secció 6.3.5) i es tornarà a assignar el valor 0 a `self.inSameState`.

Si no se satisfan algunes de les condicions per totes les opcions de l'estudiant, l'estat romandrà igual i `self.inSameState` s'incrementarà en 1.

6.3.3 Estat com a imatge

Les limitacions que van sorgir quan es va executar l'agent de Q Learning (les explicarem a la Secció 6.4.1 i la Secció 7.1) ens van portar a estudiar alternatives. Una de les opcions que vam contemplar, seguint el TFM d'Esteban Piacentino [26], era la de modificar l'estat a un format d'imatge per a poder utilitzar els algorismes que ens facilitava la llibreria Stable Baselines.

Amb la idea de preservar tota la implementació perquè seguís funcionant amb Q Learning i evitar fer grans canvis a l'entorn que ja teníem dissenyat i implementat es va decidir mantenir la mateixa estructura interna. D'aquesta manera la variable `self.state` seguia guardant l'estat tant si s'executa amb imatge o sense, però amb la diferència que una funció generaria la imatge per l'agent a la sortida de la funció `step(self, action)`.

La variable que ens indica si hem de generar la imatge és el booleà `self.withImage` i la funció que ens la generarà és la funció `self._imageStateGeneration()`.

Aquesta funció és simple i bàsicament ens passa la matriu `self.state` a una imatge quadrada de mida `self.stateImageSize` × `self.stateImageSize` representada amb una matriu quadrada de Numpy inicialitzada a 0 amb la mida que hem dit i on cada casella és un `uint8`.

Aquesta mida la trobem calculant quants bytes necessitem, aplicant la següent fórmula:

$$bytes\_necessaris = numero\_opcions * numero\_estudiants \times \left(\frac{numero\_projectes}{256} + 1\right)$$

Una vegada tenim els bytes necessaris podem obtenir la mida de la imatge aplicant la següent fórmula:

$$self.stateImageSize = 2^{\log_4 bytes\_necessaris}$$

Una vegada tenim inicialitzada la matriu, recorrem linealment `self.state` copiant valor a la nova matriu imatge. Com que hem de transformar els `int` a `uint8`, cada valor ocuparà quatre caselles de la matriu imatge i, a més, com el nou tipus no té negatius, ens caldrà sumar-li 1 abans de copiar-lo.

Una vegada hem copiat tots els valors, executarem dues funcions de Numpy, la funció `np.reshape(imageState, (self.stateImageSize, self.stateImageSize))` i també `np.stack((imageState,) * 3, axis=-1)` per assegurar-nos que la imatge manté les dimensions inicials.

D'aquesta manera ja hem generat la imatge necessària per a poder passar l'estat a l'agent d'Stable Baselines.

6.3.4 Càlcul de la recompensa (reward)

El càlcul del reward és una de les parts fonamentals de l'entorn i l'executarem utilitzant la funció `self._rewardCalculation(studentNumber, projectNumber, option)` on li passarem com a parametre l'ID de l'estudiant, del projecte i el número de l'opció. En el

nostre cas per calcular-ho utilitzem les preguntes de preferències realitzades a estudiants i empreses de l'apartat anterior.

L'estudiant té un conjunt de preferències $\mathbf{Ps_k}$, les quals cada una és un parell de valors (i, sps), on $i \in \{0, 1, \ldots, 5\}$ és la importància de la preferència per l'estudiant i $sps \in [0, 1]$ és la puntuació, un valor real que s'obté d'aplicar el mètode de càlcul específic de cada preferència. Es pot veure el resum a la Taula 12.

Preferència segons estudiant, $\mathbf{Ps_k}$	**Càlcul de la puntuació, *sps***
Distància de casa	Si la distància és menor a 50km, $\frac{50-Distancia}{50}$; sinó 0
Remot o presencial	Si coincideix 1; sinó 0
Salari	Si el salari que vol l'estudiant és menor o igual al del projecte, 1; sino si la diferència és menor a 500€, $\frac{Diferencia}{500}$; sino 0
Tipus de pràctiques	Si coincideix 1; sinó 0

Taula 12: Càlcul de la puntuació per cada pregunta a estudiants. Elaboració pròpia.

Per obtenir la puntuació final de la preferència, $fsps$ caldrà aplicar la següent fórmula:

$$fsps = \frac{i}{5} \cdot sps$$

Al vector **Fsps** s'agruparan totes les puntuacions finals de les preferències de l'estudiant.

Així mateix l'empresa també té les seves preferències $\mathbf{Pp_k}$, cadascuna formada pel mateix parell de valors (i, spp). Cada puntuació spp de preferència té el seu mètode de càlcul específic que es pot veure a la Taula 13.

Per obtenir la puntuació final de la preferència, $fspp$ caldrà aplicar la següent fórmula:

$$fspp = \frac{i}{5} \cdot spp$$

A més a més d'aquestes preguntes, el projecte també s'identifica per mitjà de tres preferències més: l'experiència laboral i de voluntariat de l'estudiant i les llengües que en domina. Cadascuna d'aquestes preferències és una llista de parells de la forma (i, spp). La spp és 1 si l'estudiant té aquesta experiència laboral, de voluntariat o parla aquesta llengua i 0 si no és el cas.

Preferència del projecte, $\mathbf{Ps_k}$	**Càlcul de la puntuació *spp***
Edat de l'estudiant	Si la diferència entre l'estudiant i la que vol el projecte és menor a 5 anys, $\frac{Diferencia}{5}$; sinó 0
Número d'any del grau	Si la diferència entre l'any de l'estudiant i el que vol el projecte és menor a 2 anys, $\frac{Diferencia}{2}$; sinó 0
Centre d'estudis de l'estudiant	Si coincideix 1; sinó 0
Nota mitjana	Si la nota de l'estudiant és menor a la que vol el projecte per menys d'1 punt: $\frac{Diferencia}{1}$; si és major d'1 punt, 0; si és millor la nota de l'estudiant, 1

Taula 13: Càlcul de la puntuacó per cada pregunta al projecte. Elaboració propia.

El valor de la variable $fspp$ per aquestes tres preferències es calcula una mica diferent, utilitzant la següent fórmula:

$$fspp = \frac{\sum_{elements\_llista}(\frac{i}{5} \cdot spp)}{numero\_elements\_llista}$$

Al vector **Fspp** hi seran totes les puntuacions finals de les preferències del projecte.

Una vegada tenim calculats els vectors **Fsps** i **Fspp**, podem obtenir la puntuació que tindrà aquesta assignació AS, la qual l'obtenim de la següent forma:

$$AS = \frac{\sum_1^{N_{\mathbf{Fsps}}} fsps}{numero\_fsps\_with\_i\_higher\_than\_0} \cdot \alpha + \frac{\sum_1^{N_{\mathbf{Fspp}}} fspp}{numero\_fspp\_with\_i\_higher\_than\_0} \cdot \beta$$

on α i β són dos valors entre 0 i 1 per equilibrar el pes de les dues puntuacions.

A més a més, per saber el reward final d'aquesta assignació, caldrà calcular l'habilitat que guanya el projecte amb l'assignació proposada. Cada projecte té un conjunt d'habilitats guardades en forma de vector **Kp** que vol que es compleixin i cada estudiant té un conjunt d'habilitats **Ks**.

Per calcular l'habilitat que el projecte aconsegueix amb aquesta assignació KS ho fem de la següent manera:

$$KS = \frac{\sum_{(kp \in \mathbf{Kp})}(if(kp\_not\_previosly\_satisfied \wedge kp \in \mathbf{Ks}) : 1; otherwise, 0)}{number\_skills\_missing} \cdot \gamma$$

on γ és un valor entre 0 i 1 per controlar el pes que tenen les habilitats sobre el total del reward.

Finalment per calcular el reward utilitzarem:

$$reward = AS + KS$$

6.3.5 Quan un estat és final?

Aquesta pregunta pot semblar trivial, però és clau per crear l'entorn i perquè els resultats on arribi l'agent siguin correctes. L'entorn és l'encarregat d'indicar-li quan és un estat final i en el nostre cas ho fem mitjançant la funció `self._isDone()`, que comprova els següents casos finals:

- **Ha assignat totes les places disponibles dels projectes per totes les opcions.** Es comprova ràpidament recorrent `self.studentsAssignedToProject` i mirant si ja ha arribat al límit de places cadascun dels projectes.
- **Ha assignat tots els estudiants per totes les opcions.** Es comprova mirant `self.assigned` i comprovant si coincideix el nombre d'estudiants assignats amb el total per cadascuna de les opcions.

Inicialment sembla que aquests dos siguin els únics casos finals en el nostre problema, però hi ha un tercer més difícil de detectar.

Aquest cas final es produeix quan ja ha assignat gran part dels estudiants, però encara hi ha projectes amb places lliures. Quan això es dóna pot donar-se al cas que cap dels estudiants que encara queden per assignar es puguin assignar als projectes restants perquè ja han sigut assignats en una de les altres opcions que tenen. Com havien definit, totes les opcions d'un estudiant han de ser diferents projectes, per tant aquí arribaríem a un estat final.

Com que és un estat que és difícil de comprovar, ja que per com hem dissenyat el sistema, hem de recórrer l'estat sencer, vam pensar en què només ho miraríem si fa molt de temps que es troba al mateix estat. Per això comprovem `self.inSameState`, que ens indica quant de temps porta al mateix estat. Ho comprovarà quan aquest valor sigui igual o

major al valor `BREAKING_EPISODE` que hem assignat per defecte a 100.000, un valor que ens permet garantir que ha solucionat gran part dels estats difícils i per tant té més possibilitats de ser final.

Aquest últim estat ho comprova recorrent `self.state` i mirant a tots els estudiants que encara els hi falta assignar algun projecte si se'ls podria assignar els que falten.

6.4 Agent

L'agent és l'encarregat de decidir quina acció tria en cada estat mitjançant algun algorisme de RL. El nostre agent funciona amb dos algorismes diferents, Q Learning i A2C.

6.4.1 Entrenament i execució utilitzant Q Learning

Q Learning és un algorisme clàssic de RL i va ser el primer que vam provar per resoldre el nostre problema. En ser un algorisme relativament senzill d'implementar vam optar per programar-lo directament.

La implementació l'hem separat en dues funcions, `_qLearningTraining(...)` que entrena l'agent i `._qLearningExecution(...)` que l'executa.

La funció d'entrenament és un bucle que s'executa `DEFAULT_TRAINING_RANGE` vegades. Per cada execució del bucle l'objectiu és entrenar l'agent ajustant la Q Table. La Q Table és una taula estadística de $numero\_estats\_possibles \times numero\_accions\_possibles$ on guarda a cada casella lo bo que és en aquell estat executar aquella acció.

Cada vegada executa el següent bucle fins que arriba a un estat final:

1. Elegeix una acció, això ho pot fer de dues maneres:
   - Elegir una acció aleatòria de l'espai d'accions.
   - Elegir l'acció amb millor puntuació per aquell estat en la Q Table.

   La forma de tria és determinarà més o menys aleatòriament, segons una constant ϵ.

2. La introdueix a l'entorn i aquest li retorna el pròxim estat, el reward i si és final.
3. Comprova la puntuació antiga *old_value* que tenia aquesta acció en aquest estat a la Q Table i la puntuació que té l'acció amb més valor del següent estat $next\_max$.
4. Obté la nova puntuació que té aquesta acció en aquest estat a la Q Table mitjançant la fórmula:

$$new\_value = (1 - \alpha) \times old\_value + \alpha \times (reward + \gamma \times next\_max)$$

   on α i γ són constants per controlar el pes que té cada part.
5. Guarda el nou valor a la Q Table.
6. Si l'estat és final acaba el bucle, sinó el repeteix amb el següent estat.

D'aquesta manera els valors de la Q Table van tornant-se cada vegada més "bons", per tant l'agent entrena.

Pel que fa a l'execució, és també un bucle que s'executa `DEFAULT_EXECUTION_RANGE` vegades el qual fa el següent procés fins que no arribi a un estat final:

1. Elegeix una acció, això ho fa elegint l'acció amb millor puntuació per aquell estat en la Q Table.
2. La introdueix a l'entorn i aquest li retorna el pròxim estat, el reward i si és final.
3. Si l'estat és final acaba el bucle, sinó el repeteix amb el següent estat.

Aquestes dues funcions permeten entrenar i executar l'agent seguint l'algorisme Q Learning, però a causa de les característiques del nostre sistema ens va sorgir un problema a l'hora d'executar-lo. Com que la Q Table és quasi sempre de grans magnituds, ja que el nombre d'estats possibles és $(numero\_projectes + 1)^{numero\_estudiants * numero\_opcions}$, això implica que per poder moure i editar-la és necessari de molta memòria RAM que no teníem a l'abast. Ens vam topar amb una limitació de *curse of dimensionality*.

Per aquesta raó vam contemplar diferents alternatives fins que vam decidir encarar el projecte mitjançant la llibreria Stable Baselines que veurem al següent apartat.

6.4.2 Entrenament i execució utilitzant Stable Baselines i AC2

La utilització de Stable Baselines permet provar nous algorismes que es troben en la seva llibreria. Per executar-los es va crear una funció que entrena i executa l'agent, `_stableBaselineTrainingAndExecution(...)`.

Per entrenar-lo, Stable Baselines ens ho ha facilitat molt, ja que simplement cal executar la funció `model.learn(total_timesteps=DEFAULT_TRAINING_RANGE)` per entrenar l'agent `DEFAULT_TRAINING_RANGE` vegades.

La implementació de l'execució tampoc és llarga. Simplement vam crear un bucle igual que per Q Learning que executa `DEFAULT_EXECUTION_RANGE` vegades l'agent. Cada execució té les següents línies de codi:

```
while not done:
action, _state = model.predict(state)
state, reward, done, info = env.step(action)
```

Llista 6.3: Execució de QLearning utiltitzant Stable Baselines.

Com es pot veure, entra en un bucle que acabarà quan l'estat sigui final i amb la funció d'Stable Baselines `model.predict(state)` obtenim l'acció que elegirà l'agent. D'aquesta manera i com l'entorn ja retorna un estat en forma d'imatge com requereix aquesta llibreria aconseguim que l'agent funcioni correctament i trobi resultats. Exposarem els seus resultats a la Secció 7.2.

6.4.3 Assignar només els millors estudiants

Fent proves amb l'agent, vam detectar que quan tenia molt poques places comparades amb el nombre d'estudiants, el sistema trigava molt a donar resultats i no assignava els estudiants amb millors notes mitjanes. Per això, abans que l'agent comenci el procés d'assignació, se li introdueix únicament els n estudiants amb millor mitjana, sent n el nombre de places totals. D'aquesta manera assigna obligatòriament els estudiants amb millor mitjana.

6.4.4 Adaptar l'agent per fer la reassignació

Una vegada el sistema assigna correctament l'estudiant, calia adaptar-lo perquè fes correctament la reassignació. Per fer-ho, és necessari que el sistema garanteixi que l'estudiant amb millor nota mitjana sigui el primer a ser assignat, el segon amb millor nota sigui el segon a ser assignat i així successivament.

La primera alternativa que vam plantejar era que l'agent hagués de tornar a assignar tots els estudiants. Per garantir l'ordre que ha de complir l'agent, hem de tornar l'entorn més restrictiu i força que només l'acció que assigna l'estudiant amb millor nota retornés un canvi d'estat i reward major a 0. Seguidament ho fes pel segon estudiant i així successivament.

Aquesta primera idea provoca que l'agent no acabi de funcionar correctament, perquè al tenir unes normes inicials tan restrictives, provoca que l'agent tardi molt a trobar les primeres accions correctes i torna l'entrenament massa costos.

Per això es va encarar la reassignació d'una manera diferent on el pes important el tingués una nova funció. Aquesta nova funció, `_assignStudentsWithSelections(...)`, és l'encarregada d'intentar assignar els estudiants que pugui en ordre.

El que fa aquesta funció és, per cada estudiant en ordre de millor nota mitjana, intentar assignar-li el projecte que demana en primera opció, si no pot, el de la segona i així successivament fins a arribar a la seva última opció. Si cap opció pot ser assignada, quedarà pendent per ser assignat.

Els estudiants que faltin per assignar seran assignats per l'agent en un entorn on l'estat inicial, en comptes de no tenir assignat cap estudiant, té assignats tots els que la funció anterior ha pogut assignar. Aquesta assignació prèvia d'alguns estudiants ho fa el constructor de l'entorn mitjançant la funció `_assignStudents()`.

D'aquesta manera no cal pràcticament modificar ni l'agent ni l'entorn i s'aconsegueixen bons resultats preservant al màxim la voluntat dels estudiants.

6.4.5 Generació de l'estadística per l'anàlisi

Una vegada el sistema funcionava es va veure interessant generar una estadística dels resultats que aconseguia. D'aquesta manera es van crear les següents funcions:

- `_executionAnalysis(...)`. Analitza una execució de l'agent d'A2C mostrant les següents dades:
  - Puntuació de preferències de l'estudiant per cada estudiant, opció i episodi.
  - Puntuació de preferències del projecte per cada projecte, opció i episodi.
  - Puntuació de preferències de les habilitats per cada projecte, opció i episodi.
  - Puntuació mitjana i màxima de preferències de l'estudiant per cada estudiant i episodi.
  - Puntuació mitjana i màxima de preferències del projecte per cada projecte i episodi.
  - Puntuació mitjana i màxima de preferències de les habilitats per cada projecte i episodi.
  - Puntuació mitjana i màxima de preferències de l'estudiant per episodi.
  - Puntuació mitjana i màxima de preferències del projecte per episodi.
  - Puntuació mitjana i màxima de preferències de les habilitats total per episodi.
  - Puntuació mitjana i màxima de preferències de l'estudiant total.
  - Puntuació mitjana i màxima de preferències del projecte total.
  - Puntuació mitjana i màxima de preferències de les habilitats total.
- `_finalAnalysis(...)`. Analitza l'execució final de reassignació mostrant quin percentatge d'estudiants han sigut assignats a la seva primera opció, segona opció fins a la seva última opció.

6.5 Scripts complementaris per facilitar les proves

Per provar casos de manera ràpida i veure si el sistema funcionava correctament va ser necessari escriure uns scripts complementaris per generar de manera aleatòria els estudiants, projectes i les eleccions dels estudiants així com un script que ho executés tot conjuntament. A continuació explicarem breument cadascun d'ells.

6.5.1 Generació aleatòria d'estudiants i projectes

L'script anomenat `generateJSON.py` és l'encarregat de generar n estudiants i m projectes aleatoris i exportar-los en forma de JSON perquè el sistema els pugui assignar.

Aquest script és senzill i bàsicament genera les respostes de la Secció 6.1 per n estudiants i m projectes assignant-los a cadascun un número identificador `ID` diferent.

S'ha intentat que totes les respostes fossin més o menys realistes acotant les possibles respostes generades aleatòriament. Les respostes que requerien un nom específic s'ha generat llistes inicials d'on elegir aleatòriament, les llistes es troben a la Taula reftable:answers.

Preguntes	**Habilitats**	**Centres**	**Idiomes**	**E.Laboral**	**Voluntariat**
Opcions	C	UPC	Català	Full-Stack	Esdeveniments
	C++	UB	Castellà	Backend	ONG Social
	Python	UPF	Anglès	Frontend	Administració
	PHP	UAB	Alemany	RH	
	Golang	UOC	Francès	Enginyer	
	Docker	URV	Grec	Administrador	
	Java		Italià	Investigador	
	.NET		Rus		
	C#		Romà		
	Ruby		Hungar		
	Node.JS		Txec		
	React		Polac		
	Javascript		Arab		
	Pearl		Xinès		
	Rest API		Japonès		
	XML		Coreà		
	PostgreSQL				
	MySQL				
	MongoDB				

Taula 14: Respostes acotades. Elaboració pròpia.

6.5.2 Elecció aleatòria dels estudiants

L'script encarregat d'escollir l'ordre de preferència de les opcions de cada estudiant és `selector.py`, on simplement per cada estudiant ordena les seves opcions de manera

aleatòria. Quan finalitza el procés genera un JSON perquè sigui fàcilment importable pel sistema.

6.5.3 Tot en un sol script

L'script `runItAll.py` executa en aquest ordre els següents scripts per facilitar les proves: `generateJSON.py`, `recommender.py`, `selector.py` i un altre cop `recommender.py`.

7. Resultats

A continuació exposem els resultats que hem obtingut del sistema que hem implementat.

7.1 Q Learning

Com hem comentat a l'apartat de la implementació del Q Learning, a causa de les característiques del problema no vam poder provar amb magnituds significatives d'estudiant ni projectes, ja que ens donava error de memòria amb únicament 10 estudiants, 10 projectes i una alternativa per estudiant. Amb només aquestes mides l'espai observable ja tenia prop de 26 milions d'estats possibles i per tant ja no era computable per l'ordinador que teníem disponible.

Per aquest motiu no hem vist interessant fer un recull de més dades per analitzar els resultats.

7.2 Stable baselines

Amb la utilització d'Stable Baselines i l'algorisme A2C sí que hem pogut provar el sistema amb una major varietat de casos i conjunts d'estudiants i de projectes.

Per comprovar els resultats ens hem basat en les següents dades que extrèiem:

- **Passos per obtenir una solució.** Nombre de canvis d'estat que ha de fer l'agent

per arribar a un estat final.

- **Grau de satisfacció dels estudiants.** Aquest valor l'obtenim de comprovar per cada estudiant com de satisfet està segons les respostes del seu qüestionari. Si se li assigna un projecte que compleix totes les preferències que demanava l'estudiant tindrà una puntuació de 100%.
- **Grau de satisfacció dels projectes.** Aquest valor l'obtenim de comprovar per cada projecte com de satisfet està segons les respostes del seu qüestionari. Si se li assigna un conjunt d'estudiants que cadascun compleix totes les preferències que demanava el projecte tindrà una puntuació de 100%.
- **Grau de satisfacció de les habilitats necessàries per al projecte.** Aquest valor l'obtenim de comprovar per cada projecte quantes de les habilitats que demanava en el qüestionari s'han satisfet. Si totes les habilitats són assolides per algun dels estudiants assignats, el projecte tindrà una puntuació de 100%.

El sistema dóna resultats en tots els casos que hem provat, tant en la primera assignació com amb la reassignació aconseguint solucions satisfactòries.

Així i tot, ens hem trobat amb una gran dificultat a l'hora de determinar el bo que són els resultats, ja que no tenim una base de dades real ni tampoc cap altre sistema similar amb el qual comparar-ho.

A més, a causa de les característiques del problema, els resultats depenent molt dels estudiants i projectes que es vulguin assignar, de si els estudiants tenen unes preferències molt exigents, si no les tenen, si els projectes tenen preferències exigents, etc. Tot això complica molt més l'anàlisi.

A continuació exposem diferents proves que li hem fet al sistema.

7.2.1 Duració entrenaments

Un factor que vam voler analitzar del nostre sistema és quan d'entrenament necessitava per aconseguir resultats bons. Ho hem fet amb tres conjunts generats aleatòriament amb diferent nombre d'estudiants i projectes. La prova ha consistit a executar el sistema

per cadascun dels conjunts 50 vegades perquè generi tres opcions per estudiant després d'haver-lo entrenat diferents duracions (1.000, 10.000, 100.000 i 1.000.000 timesteps).

A la taula 15 podem veure els resultats, on mostrem per cada conjunt i entrenament la mitjana de passos per obtenir solucions, la mitjana d'estudiants assignats, la mitjana del grau de satisfacció dels estudiants, dels projectes i de les habilitats que demanava cada projecte i els seus màxims absoluts.

	10^3			10^4			10^5			10^6		
	C1	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
#estudiants	100	100	100	100	100	100	100	100	100	100	100	100
#projectes	40	20	10	40	20	10	40	20	10	40	20	10
#places	222	105	49	222	105	49	222	105	49	222	105	49
Temps entrenament(s)	2,7	3,1	2,0	25	25	18,6	256	237	171,4	2677	2406	1935
Finalitzacions entrenament	1	0	0	8	2	4	77	13	29	830	46	130
Execucions	50	50	50	50	50	50	50	50	50	50	50	50
Mitjana passos	1214	4825	1106	1291	4772	1414	1211	4601	1381	1168	5081	1265
Mit. assignats	100	100	46,5	100	100	47,5	100	100	48,6	100	100	48,7
Mit. estud. (%)	58,3	60,0	49,5	57,3	59,9	53,4	58,7	59,1	52,5	59,2	59,4	53,6
Mit. proj. (%)	30,8	34,4	27,1	29,4	34,9	27,3	29,0	34,7	27,1	29,4	34,9	28,6
Mit. habi. (%)	32,8	55,6	64,5	32,3	56,9	64,4	32,3	58,0	61,8	32,3	53,1	61,4
Max estud. (%)	76,8	77,4	70,5	76,4	77,2	72,7	76,9	76,6	73,1	78,2	76,7	80,2
Max proj. (%)	41,5	41,6	33,7	41,5	42,4	34,9	42,0	41,8	33,7	42,3	41,9	33,6
Max habi. (%)	61,4	77,2	85,6	59,9	79,2	85,9	62,3	78,4	83,7	63,3	75,9	86,8

Taula 15: Comparació entrenament curt i llarg. Elaboració pròpia.

Com es pot veure en la gràfica, a mesura que augmentem la duració de l'entrenament, els resultats són millors, ja que millorem totes les mitjanes i reduïm el nombre de passos per arribar a una solució. A més, si el conjunt de projectes no té suficients places per tots els estudiants, a mesura que l'entrenament és més llarg, més places aconsegueix assignar.

Així i tot, la millora és escassa comparada amb el temps necessari per executar l'entrenament, per tant no creiem que ens sigui necessari estendre tant l'entrenament, ja que ràpidament el sistema obté resultats bons.

També és interessant destacar que, com ens pensàvem, els resultats són bastant diferents

depenent del conjunt que utilitzem, per tant és complicat determinar com és de bona la solució que ha trobat.

7.2.2 Proporció d'estudiants i projectes

Com hem comentat, hem detectat que el sistema obte resultats molt dispersos segons la proporció d'estudiants i projectes. Per això en aquest apartats ho hem analitzat amb més profunditat.

En la següent prova hem executat el sistema 50 vegades amb quatre conjunts diferents d'estudiants i un entrenament de 10.000 timesteps i 3 alternatives per estudiant. Hem obtingut els resultats de la taula 16

	C4	**C5**	**C6**	**C7**
#estudiants	100	100	100	100
#projectes	50	20	10	5
#places	266	103	55	33
Temps entrenament(s)	28,1	25,9	19,56	18,89
Finalitzacions entrenament	7	2	5	18
Execucions	50	50	50	50
Mitjana passos	1226	6227	1322	555
Mitjana Assignats	100	100	52,9	30,5
Mitjana estudiant (%)	58,4	59,8	53,16	39,6
Mitjana projecte (%)	29,4	34,5	29,8	32,3
Mitjana habilitats (%)	30,2	55,8	61,7	78,0
Màxim estudiant (%)	76,2	76,8	76,4	63,8
Màxim projecte (%)	43,2	41,7	36,4	39,3
Màxim habilitats (%)	56,3	81,1	80,4	97,7

Taula 16: Comparació segons la proporció d'estudiants i projectes. Elaboració pròpia.

Com podem extreure dels resultats, no podem veure que el temps d'entrenament ni la mitjana de passos estigui gaire relacionat amb la proporció de projectes i estudiants, sino molt més amb les caracteristiques dels conjunts, com són les preferències dels estudiants o dels projectes.

Tot i això, és força interessant veure que tant en el conjunt C2 i C4, on hi ha casi exactes

places disponibles que nombre d'estudiants, el sistema té una mitjana de passos molt superior a la resta de proporcions. Això es pot produir degut a que com que no omple tots els projectes ni tampoc assigna tots els estudiants, les accions que pot fer per arribar a un estat final es redueixen i per tant tarda més a trobar-les.

7.2.3 Reassignació

El sistema final també computa la reassignació amb bastant d'èxit. Igual que ens passa amb la primera assignació, és difícil determinar el bo que són els resultats perquè canvia molt depenent de l'elecció que faci cada estudiant, de les opcions que tingui cadascun i de les preferències dels estudiants i projectes.

Hem provat la reassignació amb quatre conjunts d'estudiants i projectes diferents i un entrenament de 10.000 timesteps i 3 alternatives per estudiant, obtenint les següents estadístiques que exposem a la taula 17.

	C8	**C9**	**C10**	**C11**
#estudiants	100	100	100	100
#projectes	50	20	10	5
#places	280	102	64	22
Execucions	50	50	50	50
Passos	1590	7427	3088	455
Estudiants assignats 1a assignació (%)	100	100	64	20
Estudiants assignats 2a assignació (%)	100	100	64	22
Estudiants assignats a la seva 1a opció (%)	98	82	54	16
Estudiants assignats a la seva 2a opció (%)	2	10	5	1
Estudiants assignats a la seva 3a opció (%)	0	5	2	2
Estudiants assignats a una altra opció (%)	0	3	3	3
Mitjana estudiant (%)	58,3	55,6	54,3	55,1
Mitjana projecte (%)	29,4	35,3	31,1	31,6
Mitjana habilitats (%)	29,5	55,6	71,4	65,4

Taula 17: Resultats reassignació. Elaboració pròpia.

Com es pot veure a la taula, el sistema aconsegueix bastant d'èxit, ja que sempre aconsegueix assignar totes les places disponibles i inclús quan hi ha molt poques places comparat amb el nombre d'alumnes, el sistema assigna la majoria a les primeres opcions

dels estudiants.

8. Conclusions

Hem dissenyat i implementat un sistema per assignar estudiants a pràctiques d'empresa mitjançant un algorisme de Reinforcement Learning i a més també el sistema és capaç de reajustar els resultats amb una reassignació segons les preferències que hagin dit els estudiants.

Tot el sistema s'ha dissenyat en Python i l'entorn ha sigut implementat utilitzant la llibreria Gym d'OpenAI i és funcional per qualsevol algorisme de RL que es vulgui utilitzar.

Pel que fa a l'agent, durant el transcurs del projecte es van provar dues alternatives. Inicialment es va implementar un agent utilitzant Q Learning, però a causa de les grans dimensions de l'espai observable del problema el nostre ordinador no el podia computar.

L'alternativa va ser utilitzar la llibreria Stable Baselines que ens permet utilitzar algorismes més complexos. Per poder-la utilitzar vam necessitar canviar la manera en la qual l'agent mostrava l'estat, convertint-lo en una imatge quadrada segons la quantitat d'estudiants, projectes i alternatives per estudiant que es volia. D'aquesta manera el vam poder utilitzar l'algorisme A2C per resoldre el problema.

A més de computar una primera assignació d'estudiants a projectes amb tantes alternatives per estudiant com es vulgui de manera satisfactòria, també es va adaptar el sistema perquè a més, després que els estudiants facin saber l'ordre de preferència de les seves alternatives proposades pel sistema, fos capaç d'ajustar els resultats inicials satisfent al màxim possible els desitjos de cada estudiant, donant la prioritat als estudiants amb millor nota mitjana.

El sistema implementat és capaç de donar solucions tant en la primera assignació com en la segona, així i tot també hi ha moltes millores que se li poden fer.

9. Treball futur

Una vegada conclòs aquest treball de final de grau veiem interessant proposar diferents maneres de continuar aquest projecte i donar-li continuïtat en un futur.

- **Adaptar el sistema a altres sectors.**
  El nostre projecte finalment es va orientar molt cap al sector d'IT, però seria interessant adaptar-lo perquè funciones contra una major varietat de graus universitaris i empreses.
- **Provar el sistema contra una base de dades real.**
  Seria interessant provar el sistema contra una base de dades reals i poder comprovar quan de bones són les solucions.
- **Optimitzar i quantificar el sistema.**
  Els models que hem utilitzat en aquest projecte es podrien quantificar i optimitzar per obtenir un millor rendiment. La quantització és el procés de fer càlculs utilitzant representacions de nombres de bits menors. L'optimització és el procés de podar el gràfic de càlcul de certs nodes que no afecten sobre manera la precisió.
- **Provar més algorismes de RL.**
  El ventall d'opcions dins del RL és molt gran i al tenir un entorn totalment funcional és possible provar-lo amb altres algorismes per comparar els resultats amb l'A2C que hem utilitzat per al nostre sistema.
- **Provar de reduir l'espai observable per poder utilitzar Q Learning.**
  Una opció que teníem durant el transcurs del treball i que al final vam descartar va ser intentar reduir l'espai observable, analitzant-lo i intentar treure-hi els estats impossibles. D'aquesta manera és probable que es pugui reduir significativament la mida de l'espai i poder així executar Q Learning.

Bibliografia

[1] GEORGARA, Athina; SIERRA, Carles; RODRÍGUEZ-AGUILAR, Juan A. (Maig 2020). ***TAIP: an anytime algorithm for allocating student teams to internship programs***. Paper publicat a arXiv.org. Últim accés a 19/10/2020. Disponible a internet, URL: https://bit.ly/3oT9Bce.

[2] GEORGARA, Athina; SIERRA, Carles; RODRÍGUEZ-AGUILAR, Juan A. (Juliol 2020). ***Edu2Com: an anytime algorithm to form student teams in companies***. Paper publicat en el marc del workshop AI for Social Good 2020. Últim accés a 19/10/2020. Disponible a internet, URL: https://bit.ly/2LFqcSu.

[3] **Arxiu web de la pàgina oficial d'ILOG**. Últim accés a 19/10/2020. URL: https://bit.ly/39E5nid.

[4] IBM (2009). ***IBM Completes Acquisition of ILOG [Online]***. Nota de prensa d'IBM publicada a la seva pàgina web. Últim accés a 19/10/2020. Disponible a internet, URL: https://ibm.co/2M2ECM7.

[5] **IBM CPLEX Optimizer**. Últim accés a 19/10/2020. URL: https://ibm.co/39DpFs5.

[6] WANG, Ruye (2015). ***The Simplex Algorithm*** **[Online]**. Últim accés a 19/10/2020. Disponible a internet, URL: https://bit.ly/3nRyxiU.

[7] CAPPART Quentin, MOISAN Thierry, ROUSSEAU Louis-Martin, PRÉMONT-SCHWARZ Isabeau, CIRE Andre (Juny 2020). ***Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization***. Paper publicat a arXiv.org. Últim accés a 19/10/2020. Disponible a internet, URL: https://bit.ly/39HSxzl.

[8] **GitHub**. Últim accés a 5/10/2020. URL: https://www.github.com/.

[9] **JetBains PyCharm Community Edition**. Últim accés a 5/10/2020. URL: https://bit.ly/3sAJV6e.

[10] **Python**. Últim accés a 5/10/2020. URL: https://www.python.org/.

[11] **Overleaf**. Últim accés a 31/12/2020. URL: https://www.overleaf.com/.

[12] **Tomsplanner**. Últim accés a 1/1/2021. URL: https://www.tomsplanner.es/.

[13] **GlassDoors**. Últim accés a 9/10/2020. URL: https://glassdoor.es/.

[14] **HP**. Últim accés a 9/10/2020. URL: https://hp.com/.

[15] ZHANG, Kaiqing; YANG, Zhuoran; BASAR, Tamer (Novembre 2019). ***Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms***. Paper publicat a arXiv.org. Últim accés a 19/10/2020. Disponible a internet, URL: https://bit.ly/3oWDWH0.

[16] **Estadístiques GitHub**. Últim accés a 1/1/2021. URL: https://bit.ly/35OmbCd

[17] **Documentació d'OpenAI Gym**. Últim accés a 1/1/2021. URL: https://bit.ly/3ilPukl.

[18] **Documentació d'Stable Baselines**. Últim accés a 1/1/2021. URL: https://bit.ly/3inC33x.

[19] **Pàgina de GitHub d'OpenAI Baselines**. Últim accés a 1/1/2021. URL: https://bit.ly/2LFqrNo.

[20] RAFFIN, Antonin (Agost 2018). ***Stable Baselines: a Fork of OpenAI Baselines — Reinforcement Learning Made Easy***. Article publicat a Medium. Últim accés a 1/1/2021. URL: https://bit.ly/3oT24u0

[21] MNIH, Volodymyr; PUIGDOMÈNECH BADIA, Adrià; MIRZA, Mehdi; GRAVES, Alex; HARLEY, Tim; P. LILLICRAP, Timothy; SILVER, David; KAVUKCUOGLU, Koray (Juny 2016). ***Asynchronous Methods for Deep Reinforcement Learning***. Paper publicat a arXiv.org. Últim accés a 1/1/2021. URL: https://bit.ly/2XRn1JG

[22] ALTMAN, N.S. (Juny 1991). ***AAn Introduction to Kernel and Nearest Neighbor Nonparametric Regression*** . Paper pde Cornell University disponible al seu portal web. Últim accés a 1/1/2021. URL: https://bit.ly/3qu1pzo

[23] **Pàgina de Numpy**. Últim accés a 1/1/2021. URL: https://numpy.org/.

[24] **Pàgina de Statistics**. Últim accés a 1/1/2021. URL: https://bit.ly/39GZEs4.

[25] **Pàgina de Pandas**. Últim accés a 1/1/2021. URL: https://pandas.pydata.org/.

[26] PIACENTINO, Esteban (Septembre 2019). ***Generative Adversarial Network based Machine for Fake Data Generation***. TFM de l'ETSEIB supervisat per Cecilio Angulo.