Treball de Fi de Màster

Màster Universitari en Enginyeria Industrial (MUEI)

Resolució d'un problema de planificació dinàmica de treballs mitjançant aprenentatge per reforç

MEMÒRIA

12 de setembre de 2022

Autor: Marcel Hernàndez i Camp

Director: Cecilio Angulo Bahon

Convocatòria: Primavera curs 2021 - 2022

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Resum

El problema *Job Shop Scheduling*, en el que s'ha de determinar l'ordre o seqüència òptima per a processar una sèrie de treballs en una sèrie de màquines, ha rebut una gran atenció en el món de l'organització industrial. Multitud d'heurístiques i models s'han anat proposant per tal de resoldre'l. Aquests algorismes però, acostumen a proposar solucions al problema simplificat, resolent-lo amb una sèrie de restriccions i limitacions que fan que la seva implementació sigui limitada. A més a més, la seva utilitat es veu reduïda quan es considera el problema dinàmic i no estàtic, considerant esdeveniments estocàstics com la fallada d'una màquina o l'arribada d'un nou treball. L'ús de models matemàtics exactes deixa de ser una proposta factible quan les dimensions del problema (número de màquines i treballs) augmenten.

En aquest document es proposa una solució al problema d'assignació de treballs a màquines en un entorn dinàmic, on són presents esdeveniments estocàstics com la fallada o aturada d'una màquina i el temps d'arribada dels treballs a la zona de producció. Aquest problema s'aborda mitjançant l'Aprenentatge per Reforç (*Reinforcement Learning*), una branca de la intel·ligència artificial basada en la prova i error i l'experiència.

Nomenclatura

IA	Intel·ligència Artificial
RL	Reinforcement Learning
JSSP	Job Shop Scheduling Problem
DJSSP	Dynamic Job Shop Scheduling Problem
GA	Genetic Algorithm
SA	Simulated Annealing
ACO	Ant Colony Optimization
PS	Particle Swarm Optimization
AIS	Artificial Immune System
ML	Machine Learning
MP	Markov Process
MDP	Markov Decision Process

Notació

$\doteq$	igualtat verdadera per definició
$\approx$	aproximadament igual
$\Pr\{X = x\}$	probabilitat que la variable X prengui el valor x
$\mathbb{E}[X]$	valor esperat d'una variable aleatòria X
$\mathbb{R}$	conjunt de nombres reals
$\arg\max_a f(a)$	valor de a pel qual la funció $f(a)$ pren el valor màxim
$(a, b]$	interval real entre a i b que inclou b però no a
ε	probabilitat de prendre una acció aleatòria en una política ε-$greedy$
α	paràmetre d'actualització (*step size*)
γ	factor de descompte
s, s'	estats
a, a'	accions
r	recompensa
$\mathcal{S}$	conjunt d'estats
$\mathcal{A}(s)$	conjunt d'accions possibles a l'estat s
$\mathcal{R}$	conjunt de possibles recompenses pertanyent a $\mathbb{R}$
$\in$	pertanyent a
$\lvert\mathcal{S}\rvert$	nombre d'elements del conjunt $\mathcal{S}$
$\lvert\mathcal{A}(s)\rvert$	nombre d'elements del conjunt $\mathcal{A}(s)$
t	pas de temps discret, iteració
T	temps final d'un episodi
A_t	acció presa a l'instant t
S_t	estat a l'instant t
R_t	recompensa rebuda a l'instant t
π	política, regles de comportament per a la presa de decisions
$\pi(s)$	acció presa sota una política determinista
$\pi(s, a)$	probabilitat de prendre l'acció a sota una política estocàstica
G_t	retorn a l'instant t
$p(s', r \mid s, a)$	probabilitat de trasicionar a l'estat s' amb una recompensa r donat l'estat s i l'acció a
$p(s' \mid s, a)$	probabilitat de trasicionar a l'estat s' donat l'estat s i l'acció a
$r(s, a)$	recompensa immediata al prendre l'acció a des de l'estat s
$r(s, a, s')$	recompensa immediata al prendre l'acció a des de l'estat s i transicionant a l'estat s'
$v_\pi(s)$	valor de l'estat s sota una política π; retorn esperat
$v_*(s)$	valor de l'estat s sota una política òptima
$q_\pi(s, a)$	valor de prendre l'acció a des de l'estat s sota una política π
$q_*(s, a)$	valor de prendre l'acció a des de l'estat s sota una política òptima

Índex

Índex de figures

Índex de taules

Índex d'algorismes

1 Prefaci

1.1 Origen del projecte i motivació

El problema d'assignació de recursos o treballs a màquines s'ha estudiat durant el grau (GETI) i el màster (MUEI), en diferents assignatures, fent ús de diferents mètodes. Tot i així, sempre s'ha treballat la versió estàtica i mai la dinàmica. Per tant, treballar amb el problema dinàmic, contemplant esdeveniments estocàstics, suposa anar un pas més enllà del que s'ha treballat fins al moment.

La motivació en el desenvolupament d'aquest projecte rau en l'interès respecte el món de la intel·ligència artificial i, en concret, l'aprenentatge per reforç, desconegut fins al moment d'iniciar aquest treball. És per això, que es planteja com un repte l'ús de IA per resoldre el problema dinàmic.

1.2 Requeriments previs

El propi document conté la documentació i explicació necessària per entendre el marc conceptual del problema a resoldre. També es proporciona una introducció a l'aprenentatge per reforç. Es dedica especial atenció a la Secció 4 per tal que el lector sense coneixements en intel·ligència artificial sigui capaç de seguir i entendre la solució proposada. Es presenten diferents equacions que requereixen d'una certa familiarització en notació matemàtica.

En cas de voler consultar i comprendre el codi, seran necessaris coneixements en el llenguatge de programació Pyhton i les llibreries usades.

2 Introducció

2.1 Objectius

En aquest projecte es persegueixen diversos objectius. El primer d'ells és proposar una solució al problema *Job Shop* (que es presenta amb detall a la Secció 3) des de la vessant dinàmica. Com s'ha comentat, el més usual és treballar la versió estàtica d'aquest conegut problema d'assignació de treballs a màquines. Per tant, es pretén estudiar les característiques i possibilitats de la versió dinàmica i analitzar les diferents solucions proposades amb les limitacions que presenten.

El segon és trobar una solució mitjançant aprenentatge per reforç. Fins al moment d'iniciar aquest projecte, l'RL és un àmbit desconegut i, per tant, es presenta com un repte l'ús d'aquesta branca de la IA.

S'estudiarà la teoria de l'aprenentatge per reforç i alguns dels mètodes més coneguts com a introducció a aquest món. No s'implementaran però, tots els mètodes descrits.

2.2 Abast del projecte

El problema *Job Shop* té moltes subvariants i no és viable desenvolupar un model flexible que permeti modelitzar qualsevol de les variants o multitud d'elles. La implementació serà del problema amb les característiques descrites. Es tractarà i s'abordarà el problema de manera genèrica, sense especificar-ne un cas concret d'aplicació en una indústria o línia de producció al món real.

Queda fora de l'abast el disseny d'una interfície que permeti interactuar i experimentar amb l'algorisme proposat i els seus resultats. Tot i així, a la Secció 6.3 es presenten alguns resultats de simulacions de planificació de manera visual i intuïtiva.

Tampoc s'explica el codi de programació del model en detall, escrit en llenguatge Python i que el lector pot consultar al repositori [34].

3 El problema d'assignació de treballs

3.1 El Job Shop Scheduling Problem

El *Job Shop Scheduling Problem*, d'ara endavant JSSP, és un problema d'assignació de treballs a màquines, àmpliament estudiat en l'àmbit de l'organització industrial. Representa un tipus de sistema productiu en el qual els treballs a realitzar consten d'una sèrie d'operacions. Per processar-les, es compta amb un número determinat de màquines i cada una d'elles és capaç de realitzar un conjunt determinat d'operacions. L'objectiu és determinar la seqüencia d'operacions a fer a cada una de les màquines de la manera més adient en funció del criteri o indicador d'optimització desitjat. Aquests indicadors poden ser molt diversos però es poden agrupar en tres grans grups. A continuació alguns exemples:

- Indicadors de temps
  - Temps d'entrega dels treballs
  - Temps de finalització de la producció
  - Temps mitjà de processament
- Indicadors econòmics
  - Cost de producció
  - Penalització per retard en l'entrega
  - Cost d'estoc
- Indicadors de procés
  - Disponibilitat de recursos
  - Productivitat
  - Temps morts no productius
  - Canvis d'operació en una màquina

Aquests són només alguns dels indicadors que es poden usar per tal d'avaluar una solució. És possible tenir en compte més d'un indicador a l'hora de determinar una planificació, però no tots.

El JSSP presenta moltes variants en funció de les característiques del sistema productiu que es considerin. De vegades es tracta el problema descrit amb petites variacions: els treballs consten d'una única operació, les operacions no són seqüencials, es consideren temps de transport entre màquines o de preparació, etcètera.

No és viable presentar una solució que generalitzi qualsevol d'aquestes variants i per a qualsevol dels objectius o criteris objectiu descrits. A la Secció 5 es detallen les característiques del problema amb el que s'ha treballat en aquest projecte.

En el JSSP estàtic es considera que tota la informació per a fer la programació de les tasques del període és coneguda i invariant. Així doncs, es treballa sota la hipòtesi que es coneixen les disponibilitats de les màquines, el nombre de treballs i la seva arribada i el temps de processament amb exactitud. La planificació es pot realitzar a l'inici del període i aquesta serà vàlida, ja que les condicions no variaran.

3.2 El Dynamic Job Shop Scheduling Problem

En un entorn real però, existeixen esdeveniments imprevisibles i inevitables que poden variar l'estat del sistema. En aquests casos la planificació feta sota la hipòtesi d'un entorn i condicions estàtiques deixa de tenir validesa i és necessària una re-planificació cada cop que hi ha un nou esdeveniment estocàstic. Un exemple ben clar seria la fallada d'una màquina a mitja producció. Quan això passa, totes aquelles operacions que estaven previstes en aquella màquina com es faran? Si les operacions estan restringides per unes condicions de precedència o dates límit, per exemple, la re-planificació no és tan fàcil com repartir-les entre les altres màquines disponibles, afegint-les a la cua. És sota aquest marc d'esdeveniments incerts que es presenta el *Dynamic Job Shop Scheduling Problem*; d'ara endavant DJSSP.

Els esdeveniments estocàstics que es poden considerar són diversos i nombrosos. Els més usuals:

- Relacionats amb les operacions
  - Temps de processament no conegut amb certesa
  - Arribada aleatòria de treballs
  - La data d'entrega varia
  - La prioritat o ordre de processament dels treballs canvia
- Relacionats amb les màquines
  - Fallada d'una màquina o no disponibilitat
  - Temps de processament d'una màquina variable
- Relacionats amb el procés
  - El departament de qualitat rebutja un treball
  - Producció inestable

- Altres esdeveniments
  - Absència de l'operari de la màquina
  - Problemes amb la matèria prima

3.3 Estat de l'art

3.3.1 Mètodes exactes

El problemes de planificació de treballs solen ser problemes NP, és a dir, que el temps per resoldre'ls de manera exacta no és polinòmic, sinó exponencial [1]. En el JSSP, a mesura que augmenta la dimensió del problema (número de màquines i treballs), les possibles combinacions augmenten de manera exponencial, fent-ho també el temps necessari per trobar una solució exacta. El mètode de Johnson, per exemple, garanteix la solució òptima en la minimització del temps de processament de les màquines per a versions del problema simplificat: N jobs i fins a M=3 màquines. Aquest mètode però, només garanteix solucions sota una serie d'hipòtesis i condicions que restringeixen considerablement la seva aplicació en casos reals [3]. Quan el problema incorpora esdeveniments estocàstics (DJSSP), aquests mètodes ja no són vàlids.

Per tant, els mètodes exactes només poden proporcionar solucions al problema estàtic i sota una sèrie de condicions i hipòtesis. El seu ús no és factible si es considera el problema dinàmic.

3.3.2 Mètodes aproximats

D'altra banda hi ha algorismes que no poden garantir la solució òptima, però sí una bona aproximació en un temps computacional més acotat. Alguns autors van proposar modificacions al model de Johnson per tal de generalitzar l'heurística al cas de N jobs i M màquines [2]. Existeixen multitud de models i algorismes que presenten aproximacions al JSSP estàtic però, donat que el cas d'estudi d'aquest projecte és el problema dinàmic, només es farà un repàs de la literatura, i es comentaran les tècniques més usades, per abordar el DJSSP.

Primer de tot, cal diferenciar dos tipus d'estratègies a l'hora de resoldre una planificació de treballs dinàmica.

Planificació totalment reactiva
És caracteritza per no elaborar una planificació prèvia inicial, sinó que se'n genera una de nova cada cop que succeeix un esdeveniment estocàstic. Quan es genera una nova planificació, es fa sense considerar possibles esdeveniments estocàstics futurs. La re-planificació és constant i això implica que sovint el cost augmenti.

Planificació predictiva i reactiva
Una altra estratègia és realitzar una pre-planificació sense considerar esdeveniments estocàstics futurs. Aquesta serveix de referent i, cada cop que un esdeveniment fa inviable la seva aplicació,

es modifica l'actual marc per tal d'adaptar-lo a les noves condicions i optimitzar-ne els resultats. A grans trets, en comptes de generar una nova solució des de zero, es modifica l'actual.

A continuació es presenten algunes de les tècniques i mètodes més usats en la resolució del DJSSP [4]. No s'expliquen en detall, ja que seria massa extens.

1. Assignació per regla de prioritat
   És el mètode més tradicional, basat en un indicador de prioritat per a cada treball que depèn de l'objectiu. Les assignacions es realitzen de més a menys prioritari. Pot ser usat per a una programació dinàmica a temps real per la seva simplicitat, cost computacional baix i la facilitat per entendre'l i aplicar-lo. Combinat amb algorismes genètics, s'ha comprovat que dona molt bons resultats [5].

2. Expert System
   Els sistemes experts són sistemes que, a través d'intel·ligència artificial, generen heurístiques complexes que simulen el comportament i les decisions d'un humà o entitat coneixedora i experta en el tema concret. El seu desenvolupament és molt complex, llarg i costos. ISIS és el primer *expert system* per resoldre el DJSSP [5].

3. Cerca Tabú
   La cerca tabú és una meta-heurística que, partint d'una solució factible, busca solucions veïnes (solucions pràcticament idèntiques on només es varia algun detall). D'aquesta manera, es pot buscar recursivament solucions veïnes millors a l'actual. La cerca acaba quan la solució no es pot millorar. L'optimització local queda totalment condicionada per l'estructura del veïnat de solucions; sovint l'òptim local on ha convergit no és l'òptim global. És per això que aquestes tècniques d'optimització es combinen amb d'altres, com els algorismes genètics, per tal de millorar el rendiment i els resultats.

4. Genetic Algorithm (GA)
   Els algorismes genètics generen noves solucions a partir de dues solucions (d'aquí el nom "genètic"). Es van realitzant combinacions de solucions per tal de millorar el resultat. El problema principal és que en treballar amb problemes grans d'optimització combinatòria, l'espai de cerca és molt gran i el temps de càlcul elevat. Com utilitzar els algorismes genètics de manera eficient s'ha considerat un repte i àmbit d'investigació els últims anys. Christian B. i Dirk C. proposen un model per minimitzar el temps mitjà de processament basat en algorismes genètics. Aplicat a un sistema de producció estàtic o dinàmic, ha donat millors resultats que l'Assignació per Regla de Prioritat [6]

5. Simulated annealing (SA)
   El recuit simulat està inspirat en el procés de recuit aplicat a l'acer i ceràmiques: escalfar i després refredar lentament. Amb l'augment de temperatura, els àtoms augmenten la seva energia i poden desplaçar-se de les seves posicions inicials (mínim local d'energia). Amb el refredament lent, s'augmenten les probabilitats de recristal·litzar en posici-

ons amb menys energia que la inicial (mínim global). Aquest algorisme doncs, explora el veïnat de manera iterativa i, probabilísticament, decideix si transicionar a la nova solució o mantenir-se en l'actual. L'objectiu és trobar una configuració més propera a l'òptima. Per si sol no resulta ser gaire eficient, però al ser un mètode d'optimització local, es pot aplicar com a post-procés d'un altre algorisme [7].

6. Ant Colony Optimization (ACO)
   En aquest cas, es tracta d'iterar sobre un nombre determinat de solucions factibles inicials. Aquelles assignacions fetes en les solucions amb millor puntuació serà més probable que estiguin a la solució òptima [8]. Yan-hai et al. [9] proposaren un model basant en ACO però amb un preprocés (tractament de mutants) per evitar òptims locals.

7. Particle Swarm Optimization (PSO)
   És un mètode computacional que optimitza un problema de manera iterativa. Es parteix d'una població (eixam) de solucions factibles (partícules) que es mouen en cada iteració cap a l'òptim local més proper, però també cap a la millor solució trobada [10]. El seu principal defecte és la convergència prematura. Wang et al.[11] van aplicar el PSO combinat amb un algorisme genètic per resoldre el DJSSP obtenint uns resultats factibles i efectius.

8. Artificial Immune Systems (AIS)
   Els sistemes immunitaris artificials són una classe de sistemes d'aprenentatge automàtic inspirats en els principis i processos del sistema immunitari dels vertebrats. Les tècniques comunes estan inspirades en teories immunològiques concretes, que expliquen el funcionament i comportament del sistema immunitari adquirit mamífer: Algorisme de Selecció Clonal, de Selecció Negativa, de Xarxes Immunes i de Cèl·lula Dendrítica. Yu Jianjun et al. [12] van proposar un AIS per a una planificació dinàmica de tasques i van introduir altres elements innovadors.

9. Reinforcement Learning (RL)
   Tot i no ser un dels mètodes més usats, recentment està prenent molta importància. L'aprenentatge per reforç es basa en la prova i error i la presa de decisions en funció de les accions preses en el passat; l'experiència [1]. L'ús de l'RL i el Deep RL han guanyat interès en la comunitat d'IA pels seus èxits, com en els jocs Atari [14], l'AlphaGo [15] o l'AlphaStar [16]. La seva aplicació al DJSSP presenta certs avantatges. En primer lloc, és més flexible que la majoria de mètodes mencionats. En segon lloc, ofereix la possibilitat de programar de manera incremental (reactiva) sense un sobrecost afegit. Finalment, la característica més interessant és que no només aprèn a optimitzar una instància del DJSSP, sinó també a reutilitzar l'experiència adquirida en instàncies anteriors. Depenent de com es definei-

[1] A la Secció 4 s'explica amb més profunditat l'aprenentatge per reforç

xi l'entorn, és possible usar l'aprenentatge adquirit en un altre DJSSP de característiques similars.

L'aplicació de l'RL al DJSSP és prometedora, tot i que la literatura disponible acostuma a considerar un únic tipus d'element estocàstic a la vegada: l'arribada de nous treballs o el temps de processament.

Autors	**Esdeveniments Dinàmics**	**Objectiu**	**Algorisme**	**Espai d'Estats**	**Policy**
Wang et al.[18]	Arribada de treballs, Temps de processament	Makespan, Suma de retards, Temps mitjà processament	Q-learning	Discret	ε-greedy
Fonseca et al.[19]	Temps de preparació	Makespan	Q-learning	Discret	ε-greedy
Shahrabi et al.[20]	Arribada de treballs	Temps mitjà processament	Q-learning	Discret	ε-greedy
Wang et al.[21]	Arribada de treballs	Penalització per retard i precocitat	Q-learning	Discret	ε-greedy
Wang et al.[22]	Temps de processament	Penalització per precocitat, cost de completament	Double Q-learning	Discret	ε-greedy
Bouazza et al.[23]	Arribada de treballs	Makespan, Temps total de processament ponderat	Q-learning	Discret	Annealed linealry ε-greedy
Luo et al.[24]	Arribada de treballs	Retard total	DDQN	Continu	Soft-max
Luo et al.[25]	Arribada de treballs	Retard total, Rati d'utilització de les màquines	DDQN	Continu	Annealed linealry ε-greedy
Chang et al.[17]	Arribada de treballs	Penalitzacions per precocitat i retard	DDQN	Continu	Soft ε-greedy

Taula 1: Alguns dels mètodes existents de RL per resoldre la programació d'operacions dinàmica. *Font:* [17]

10. Multi-Agent System (MAS)

    Els sistemes multi-agent permeten resoldre problemes complexos. Cada un dels agents s'encarrega de fer una tasca en particular, fent ús de procediments heurístics o mètodes

descrits o bé simples funcions. Un sol agent no seria capaç de proposar una solució, donada la dimensió i/o complexitat del problema. En el cas del JSSP, Yoo i Müller [13] proposen una solució amb MAS que va més enllà del problema d'assignació de tasques i inclou funcionalitats d'aprovisionament i estoc. L'agent principal (SA) realitza la programació d'operacions a fer a les màquines, mentre que altres agents, basats en regles, s'ocupen de l'estoc i l'aprovisionament.

11. Mètode Híbrid d'Optimització
Com s'ha comentat, sovint es combinen alguns dels mètodes mencionats per tal de cobrir les flaqueses que presenta cada un dels mètodes aplicat per separat. Els mètodes híbrids han sigut, i són, un àmbit de recerca en els últims anys, amb l'objectiu de trobar l'encaix entre diferents tècniques que permeti obtenir uns millors resultats.

4 Aprenentatge per reforç

En aquesta secció es proporciona tota la informació necessària per tal de comprendre la solució proposada a la Secció 6. Tots els conceptes explicats es basen i es poden trobar al llibre de RL de referència de Richard S.Sutton i Andrew G. Barto [28]. El lector pot consultar aquest llibre per obtenir una explicació més extensa i ampliar els continguts i algorismes tractats en aquest document.

4.1 Fonamentació

El *Reinforcement Learning* (RL) és considerat una branca del *Machine Learning* (ML) o Aprenentatge Automàtic. Dins del ML hi ha dues altres categories: l'aprenentatge supervisat (*supervised learning*) i l'aprenentatge no supervisat (*unsupervised learning*).

El supervisat, el més estudiat i aplicat, entrena amb un conjunt de dades etiquetades; parelles de *input-expected output*. L'objectiu és que el model sigui capaç de generalitzar i extrapolar el coneixement a noves dades. Les tasques més comunes a fer són les de classificació i regressió i, en el món de la visió per computador, la classificació i detecció d'objectes.

El no supervisat no parteix de dades etiquetades (*expected output*), ja que té com a objectiu trobar patrons i estructures internes de les dades. Es realitzen tasques com la d'associació, agrupació i reducció de les dimensions de les dades.

L'aprenentatge per reforç no parteix de dades etiquetades i tampoc pretén trobar l'estructura interna de les dades. En canvi, es basa en aprendre què fer en cada situació per tal de maximitzar una recompensa numèrica. A l'aprenent no se li diu quines accions prendre en funció de l'estat, sinó que ho ha de descobrir i aprendre quines accions generen una recompensa major, no només a curt, sinó també a llarg termini. En la majoria de casos d'aplicació de l'RL, les accions preses no només afecten a les recompenses immediates sinó també a les futures, ja que condicionen el següent estat o situació. La prova i error i el condicionament de les recompenses futures són els trets característics de l'aprenentatge per reforç.

De fet l'aprenentatge per reforç és el tipus d'aprenentatge més semblant al dels humans i animals. La major part de les accions que prenem durant el dia a dia es basen en l'experiència prèvia, encara que no en siguem conscients. Un cas ben clar és quan aprenem a jugar a un joc de taula. Normalment no se'ns explica quin és el millor moviment o acció a fer en funció de l'estat del joc, tret de casos molt concrets. Simplement se'ns dona informació de les normes de joc i quin és l'objectiu. A base de jugar-hi, cadascú desenvolupa un comportament de joc en funció de les seves experiències passades. Aquest és precisament l'objectiu de l'aprenentatge per reforç, trobar la millor política (*policy*) de comportament que determini quina acció convé més en cada estat per maximitzar la recompensa total.

4.1.1 Elements de l'Aprenentatge per reforç

A l'RL s'hi poden identificar diversos elements essencials i característics. L'agent (*agent*) és l'entitat que adquireix el coneixement tot interaccionant amb l'entorn (*environment*). A més a més, en un sistema de RL es poden distingir quatre altres elements: una política (*policy*), un senyal de recompensa (*reward signal*), una funció de valor (*value function*) i, opcionalment, un model de l'entorn.

La *policy* és el que determina el comportament de l'agent a l'hora de prendre decisions en cada instant o estat. En alguns casos, la política pot ser una simple taula que relaciona estats amb accions i en altres pot incloure càlculs extensos. És el nucli de l'aprenentatge per reforç ja que permet determinar el comportament. Les polítiques poden ser deterministes, on l'acció escollida és sempre la mateixa en un estat determinat, o estocàstiques, especificant probabilitats per a cada acció.

El *reward signal* defineix l'objectiu de l'aprenentatge. Quan l'agent pren una acció en un estat determinat, l'entorn retorna aquesta recompensa. La política es veurà alterada segons aquesta recompensa obtinguda. A grans trets, si l'acció és beneficiosa per assolir l'objectiu es veurà reforçada; altrament, s'afeblirà .Sovint però, la definició d'aquest senyal de recompensa no és trivial i pot induir a biaixos en la solució.

La *value function* indica què és bo a llarg termini. A diferència de la recompensa, que dona informació de què és bo en aquell instant, la funció valor indica la conveniència d'un estat (o la presa d'un acció en un estat) a llarg termini. Un estat pot tenir un elevat valor tot i tenir recompenses immediates baixes. Això és degut a que els estats futurs als quals es pot accedir des d'aquest estat tenen recompenses immediates elevades. També pot succeir a l'inrevés. Els valors es construeixen a partir de les recompenses i s'utilitzen per determinar les accions a prendre. Al següent apartat es presenta la formulació matemàtica per acabar d'assentar aquests conceptes.

L'últim element és un possible *model* de l'entorn. Aquest simula el comportament de l'entorn, per exemple, predient l'estat futur al prendre una determinada acció en un cert estat.

4.2 Finite Markov Decision Processes (Finite MDP)

Un MDP és una extensió d'un procés de Markov (nom degut al matemàtic Andrey Markov [29]) . Els processos o cadenes de Markov són models estocàstics que descriuen seqüencies de possibles esdeveniments i on la probabilitat dels esdeveniments només depèn de l'estat anterior. En un MDP entren en joc les decisions que es poden prendre en cada estat. Per tant, els MDPs són processos de control estocàstics de temps discret. Són la formalització de molts problemes de RL, ja que proporcionen un marc matemàtic per a modelar la presa de decisions en situacions en què els resultats no estan totalment sota el control de qui pren les decisions, sinó que tenen una component aleatòria. En aquest cas, els *Finite* MDPs són MDPs on els espais d'estats ($\mathcal{S}$), accions ($\mathcal{A}$) i recompenses ($\mathcal{R}$) són finits.

Un MDP queda totalment descrit per la tuple d'elements (S, A, P, R, γ):

- **S** : Espai d'estats en que el sistema es pot trobar. La representació de l'estat ha d'incloure informació de tots els aspectes que afecten el futur. En cas contrari no compliria la Propietat de Markov (Equació 1).
- **A** : Espai d'accions. Representa totes les possibles accions que es poden fer en els diferents estats. Les accions poden ser les mateixes en tots els estats o ser particulars de cada un d'ells $(A(s))$.
- **P** : Probabilitats de transició des d'un estat a un altre prenent una acció concreta. (veure Equació 2)
- **R** : Recompensa o penalització obtinguda en realitzar una acció concreta en un estat de partida determinat. Aquesta pot dependre de l'estat futur $(R(s, a, s'))$ o no $(R(s, a))$. És un valor numèric.
- $\boldsymbol{\gamma}$: Factor de descompte (*discount rate*) que defineix la rellevància de les recompenses futures enfront la recompensa immediata. Aquest factor és un valor comprés entre 0 i 1. Tal i com es pot deduir de l'Equació 4, quan γ=1 les recompenses futures prenen la mateixa importància que la recompensa immediata. En canvi a mesura que γ s'aproxima a 0, les recompenses futures perden pes i la recompensa immediata és més important.

Un concepte molt important en els MDP és que els estats futurs i passats són independents; es poden realitzar prediccions sobre estats futurs únicament amb la informació de l'estat actual. Això és el que s'anomena la Propietat de Markov:

$$p(S_t, R_t \mid S_{t-1}, ..., S_1, A_{t-1}, ..., A_1) \;=\; p(S_t, R_t \mid S_{t-1}, A_{t-1}) \qquad t \geq 1 \tag{1}$$

on p és la probabilitat d'ocurrència de l'estat s' i la recompensa r a l'instant t, donat l'estat anterior s i l'acció presa a a l'estat *t-1*:

$$\begin{gathered} p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \qquad t \geq 1 \\ \forall s', s \in \mathcal{S}; \quad \forall r \in \mathcal{R}; \quad \forall a \in \mathcal{A} \end{gathered} \tag{2}$$

Interacció Agent - Entorn

Com s'ha dit, l'entitat que adquireix coneixement i pren les decisions es defineix com l'agent (*agent*). Aquest, tal i com es veu a la Figura 1, interacciona amb l'entorn (*environment*) que s'encarrega de reaccionar a les accions preses (A_t) per l'agent actualitzant l'estat (S_t) i proporcionant una recompensa (R_t).

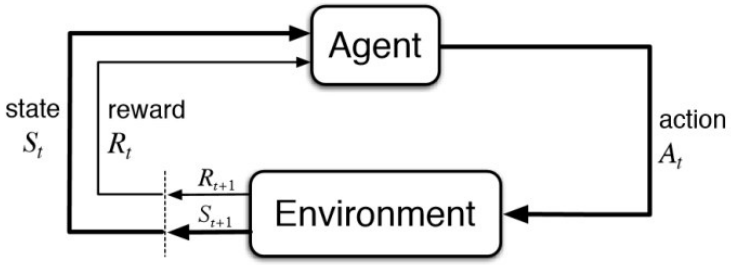

Figura 1: Representació típica d'un escenari d'aprenentatge per reforç. *Font:*[*28*]

L'agent corregeix el seu comportament a l'hora de triar accions amb aquest senyal de recompensa rebut i tria novamnet una altra acció per al nou estat. I així successivament.

Objectius i Recompenses

En l'aprenentatge per reforç, l'objectiu es formalitza mitjançant el senyal de recompensa. A cada iteració, la recompensa és un nombre real $R_t \in \mathbb{R}$. L'objectiu de l'agent no és simplement maximitzar la recompensa immediata sinó la recompensa total acumulada. El senyal de recompensa no s'ha d'entendre i fer servir com una manera de guiar l'agent a l'hora de prendre decisions. La voluntat és simplement informar a l'agent de com de bo és a curt i llarg termini el comportament que està tenint per assolir un objectiu. Per exemple, per a un agent que està aprenent a jugar a un joc de taula com els escacs, es podria definir una recompensa de +1 si guanya, una recompensa o penalització de -1 si perd i, per a totes aquelles accions que no condueixen a un estat terminal, una recompensa nul·la ($r = 0$). No s'hauria de pretendre donar recompenses positives per a aquelles accions que considerem beneficioses o recompenses negatives per a aquelles que considerem dolentes. Estaríem basant el senyal de recompensa en la nostra experiència i no en l'objectiu de l'aprenentatge, que en aquest cas seria guanyar la partida.

La recompensa ha de ser la manera de comunicar a l'agent *què* es vol aconseguir i no *com* es vol aconseguir.

Episodis i Retorns

Com s'ha comentat, l'agent provarà de maximitzar el retorn esperat (*expected return*), és a dir, el sumatori de recompenses que pot acumular des d'un cert estat en un instant t fins a l'estat final a l'instant T.

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T, \tag{3}$$

Ara bé, aquesta formulació només té sentit quan existeix una noció clara del temps final d'acabament. Aquestes divisions de l'aprenentatge en subseqüències, des de l'estat inicial s_0 a l'estat final s_T, s'anomenen episodis (*episodes*). Cada episodi acaba quan l'entorn arriba a un *estat terminal* en un instant T. Aquest estat terminal no necessàriament ha de ser el mateix en tots els

episodis, de la mateixa manera que T pot diferir entre episodis. Un cop finalitzat l'episodi, l'entorn es reinicialitza a un estat inicial (que pot ser únic o no). Aquests casos s'anomenen *episodic tasks*. Seria el cas, per exemple, d'una partida d'escacs, on hi ha diversos estats terminals (victòria, derrota i taules) i T pot ser diferent en cada partida. Un cop s'ha assolit un estat terminal, es reinicialitza l'entorn (el taulell).

D'altra banda, quan T no és identificable, sinó que continua l'aprenentatge sense límit, s'anomenen *continuing tasks*. Un robot d'assistència personal, per exemple no té un estat terminal. L'ús de la fórmula (Equació 3) per a determinar el retorn és problemàtic ja que $T \to \infty$ i el retorn G_t podria fàcilment esdevenir ∞.

Aquí és on entra en joc el paràmetre descrit γ. Fent ús d'aquest factor de descompte es pot definir el retorn esperat descomptat (*expected discounted return*) com:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{4}$$

Així doncs, la definició de l'Equació 3 esdevé un cas particular per a $\gamma = 1$. És important notar el fet que G_t es pot expressar en termes de recompensa immediata i del retorn esperat a t+1:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... = \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ...) = \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{5}$$

Amb aquesta nova definició de retorn esperat amb descompte, es decideix quin pes es dona a les recompenses futures enfront a la recompensa immediata. Encara que l'Equació 4 sigui la suma d'un nombre infinit de termes, sempre que $\gamma < 1$ i la recompensa sigui diferent de zero i constant, G_t serà un valor finit.

Polítiques i Funcions valor

La majoria d'algorismes d'aprenentatge per reforç es basen en estimar unes *value functions*. Aquestes són funcions d'estat (*state-value functions*), que estimen com de bo és per a l'agent estar en un cert estat, o bé funcions de parelles d'estat-acció (*action-value functions*), que estimen com de bo és prendre una acció en un estat donat, en termes de futures recompenses esperades. Aquestes funcions es defineixen respecte a un comportament determinat; el que s'anomenen polítiques (*policies*).

Formalment, una *policy* és un "mapeig" d'estats a probabilitats de seleccionar cadascuna de les possibles accions. L'agent pot seguir diferents polítiques a l'hora de prendre les seves decisions

i, per tant, depenent de la política seguida, els resultats seran uns o uns altres. Si l'agent segueix una política *estocàstica* π a l'instant t, aleshores $\pi(a \mid s)$ és la probabilitat que l'acció escollida sigui a si l'estat actual és s i s'està seguint la política π:

$$\pi(s \mid a) = \mathbb{P}\left[A_t = a \mid S_t = s\right], \qquad a \in \mathcal{A},\ s \in \mathcal{S} \tag{6}$$

En canvi, si la política és *determinista*, $\pi(s)$ representa l'acció a presa a l'estat s seguint aquesta π.

La *value function* d'un estat sota una política π es denota $v_\pi(s)$ i representa el retorn esperat des d'un estat s. Per un MDP podem definir la funció estat-valor (*state-value function*) v_π sota una política π com:

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \forall s \in \mathcal{S} \tag{7}$$

De manera similar es pot definir la funció acció-valor (*action-value function*) sota una política π com:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right], \forall s \in \mathcal{S}\ , \forall a \in \mathcal{A} \tag{8}$$

Aquesta última representa el retorn esperat començant a l'estat s i prenent l'acció a.

Finalment, es defineix l'equació de Bellman per v_π, que expressa la relació entre el valor d'un estat s i els valors dels seus successors estats, mitjançant una suma de totes les possibilitats ponderada per la seva possibilitat d'ocurrència ($\pi(a \mid s)p(s', r \mid s, a)$).

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] = \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\right] = \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right], \qquad \forall s \in \mathcal{S} \end{aligned} \tag{9}$$

Optimal Policies and Optimal Value Functions

Es podria considerar que trobar la solució a un problema de RL significa trobar la política que maximitza el retorn a llarg termini. Per ordenar les diferents polítiques, es fa ús de les *value functions*. Una política π es considera millor que una altra π' si el retorn esperat en la primera és major per a qualsevol $s \in \mathcal{S}$. És a dir : $\pi \geq \pi'$ si i només si $v_\pi(s) \geq v_{\pi'}(s)$ per qualsevol $s \in \mathcal{S}$.

Sempre existeix una *policy* millor o igual que totes les altres. Aquesta és la política òptima. N'hi pot haver més d'una i es representen com π_*. Totes elles comparteixen la mateixa *optimal state-value function* v_* definida com:

$$v_*(s) \doteq \max_\pi v_\pi(s), \qquad \forall s \in \mathcal{S} \tag{10}$$

Les polítiques optimes també comparteixen l'*optimal action-value function*, denotada com q_* i definida com:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \qquad \forall s \in \mathcal{S},\ \forall a \in \mathcal{A} \tag{11}$$

La funció òptima acció-valor (Equació 11) es pot reescriure en termes de la funció estat-valor:

$$q_*(s,a) = \left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \tag{12}$$

De manera intuïtiva es podria dir que la *Bellman Optimality Function* expressa el fet que el valor d'un estat sota una política òptima ha de ser igual al retorn esperat sota la millor acció en aquest estat. A continuació es mostren les equacions d'optimitat de Bellman de v_* i de q_*

$$\begin{aligned} v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned} \tag{13}$$

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s', r \mid s, a)[r + \gamma \max_{a'} q_*(s', a')] \end{aligned} \tag{14}$$

El principi d'optimitat de Bellman defineix que una política òptima té la propietat que, sigui quin sigui l'estat inicial i la decisió inicial, les decisions restants han de construir una política òptima en relació al resultat de la primera decisió ja presa. En altres paraules, independentment de l'estat en el que es trobi el sistema, la política òptima definirà el millor comportament des d'aquest estat.

Per a MDPs finits, l'equació d'optimitat de Bellman (Equació 13) té una única solució. En realitat

és un sistema d'equacions, una per a cada estat. Així doncs, si hi ha$|S|$ estats, hi ha$|S|$ equacions amb$|S|$ incògnites. Si la dinàmica (P) del sistema és totalment coneguda, es pot resoldre aquest sistema d'equacions per v_* usant qualsevol mètode de resolució d'equacions no lineals.

Una vegada es té v_*, és relativament fàcil determinar la política òptima. Per a cada estat, s'ha de fer una cerca per determinar l'acció que condueix a un estat futur amb el millor valor $v(s)$. Si s'obté q_*, determinar la política òptima és encara més fàcil. Per a cada estat s, es prendrà l'acció amb el valor $q_*(s, a)$ major.

El punt fort d'aquesta q_* és que permet seleccionar les millors accions sense tenir cap coneixement dels estats successors ni els seus valors, és a dir, sense conèixer res sobre la dinàmica del sistema.

4.3 Tècniques d'aprenentatge per reforç

En aquesta secció es dona una idea general sobre els diferents tipus d'algorismes d'aprenentatge per reforç. L'objectiu és que el lector pugui tenir un marc conceptual per tal de comprendre l'algorisme, detallat a la Secció 4.3.4.2, que serà aplicat a la Secció 6 per abordar el problema de la programació de treballs en un entorn dinàmic.

4.3.1 Tipus d'algorismes

A continuació es presenten algunes de les possibles classificacions. N'hi ha d'altres, però aquestes són importants i suficients per al propòsit d'aquest document.

Model-free vs. Model-based Methods

Aquesta primera diferenciació rau en l'existència d'un model perfecte del sistema o no. Aquest model permet conèixer sense cap mena de biaix les transicions i recompenses obtingudes en cada estat i per a cada acció. Els mètodes que parteixen d'un model perfecte del sistema s'anomenen *model-based*. També inclou aquells mètodes que proven de descobrir i crear un model del sistema a través de l'experiència. Aquests poden donar lloc a solucions òptimes respecte al model generat però sub-òptimes per al problema real. Aquest aprenentatge del model del sistema és una tasca difícil i pot no ser gaire bo i/o presentar un biaix. En aquests casos, l'agent tindrà un comportament bo respecte el model generat però un comportament sub-òptim (o dolent) en l'entorn real.

D'altra banda, els *model-free* no fan servir cap mena de model del sistema per aprendre i generar una política òptima. En canvi, el seu aprenentatge es construeix a partir de l'experiència. Mitjançant la prova i error, l'agent va descobrint com de bo és cada estat o com de bo és prendre cada acció en cada estat.

Així doncs, la diferència rau en si l'agent té accés a un model de l'entorn (o l'aprèn) o no. Depenent del problema, serà més convenient un tipus de mètode o un altre, o inclús una combinació

dels dos pot ser adient. En entorns complexos i dinàmics, l'aplicació d'un model pot ser difícil.

Offline vs. Online Methods
Aquesta segona categorització fa referència en com s'obtenen les dades o mostres per entrenar. En un mètode *online*, l'agent està interactuant amb l'entorn i va recollint dades (s, a, r, s') que va usant per aprendre i elaborar la seva política objectiu. En canvi, en un mètode *offline* l'agent ja no interactua amb el sistema sinó que usa dades recollides anteriorment i emmagatzemades. Per tant, es treballa amb un conjunt fix d'interaccions; no se'n poden recollir més.

Així doncs, dependrà de si l'agent pot seguir interactuant o no amb l'entorn.

On-Policy vs. Off-Policy Methods
Aquesta tercera classificació depèn de si la política que s'està actualitzant i millorant és la mateixa o no que la que s'esta seguint per generar dades. Els anomenats *On-Policy* proven d'avaluar i millorar la política que s'està usant per prendre decisions. En canvi, els *Off-Policy* avaluen i milloren una política diferent a la que l'agent està seguint per recol·lectar dades.

La creació d'una política òptima passa per visitar tot l'espai d'estats i avaluar-ne les possibilitats. Si l'agent segueix el mateix comportament avaricios (*greedy*) a l'hora d'entrenar, aquesta visita de l'espai d'estats es pot veure truncada. Si en les primeres iteracions s'explora una acció que fa que $q(s, a) > q(s, a') \quad \forall a' \in \mathcal{A}$ i $a' \neq a$, la resta d'estats a' es consideraran pitjors i no s'escolliran mai. Això no implica que siguin pitjors, ja que és possible que algun d'ells encara no s'hagi visitat mai i no se'n tingui informació. Una política diversa a l'hora de prendre decisions ajuda a aquesta exploració de tot l'espai d'estats. L'algorisme implementat és el Q-*learning*, un mètode *off-policy* explicat amb més detall a la Secció 4.3.4.2. D'altra banda, SARSA és un mètode *on-policy* (Secció 4.3.4.1).

Value-based vs Policy-based Methods
En aquest cas, els *value-based* són aquells mètodes que aproximen les funcions v o q i aquestes determinen la política. La política sense aquestes *value functions* no existiria. D'altra banda però, hi ha els *policy-based*, que aprenen una política parametritzada capaç de triar accions sense consultar cap funció de valor. Tanmateix, la funció de valor s'estima igualment per tal d'ajustar els paràmetres de la política. Els mètodes basats en la política són mètodes aproximats, mentre que els basats en les funcions de valor poden ser tant tabulars com aproximats.

Tabular vs. Approximate Methods
Els mètodes tabulars són aquells que emmagatzemen els valors q (o valors v) i els van actualitzant a mesura que es van generant iteracions en els diferents estats i accions. Recordem que la convergència està garantida en l'òptim si el nombre de vegades que s'ha avaluat cada parella (s, a) és infinit (o suficientment gran). Ara bé, en molts problemes l'espai d'estats i accions és combinatori i enorme i fa que sigui intractable el problema amb un mètode tabular, tant a nivell d'emmagatzematge com computacional per omplir correctament la taula. En aquests casos l'objectiu és determinar una bona aproximació amb un ús de recursos més limitat. A tal efecte,

s'opta per un mètode aproximat, que fa ús d'una funció d'aproximació que generalitza el coneixement adquirit (amb un reduït nombre d'estats i accions) per a qualsevol estat i acció. Mentre que el Q-*learning* (Secció 4.3.4.2) és un mètode tabular, que emmagatzema tots els valors $Q(s, a)$ de totes les parelles d'estats (files) i accions (columnes), el Deep Q-learning fa ús d'una xarxa neuronal que, donat un estat (*input*), retorna l'acció a prendre (*output*).

4.3.2 Programació dinàmica

La programació dinàmica (DP) fa referència a un conjunt d'algorismes que es poden utilitzar per trobar polítiques òptimes donat un model perfecte de l'entorn modelitzat com un Markov Decission Process. L'aplicació pràctica d'aquests a problemes d'aprenentatge per reforç es veu limitada per dues causes: la necessitat d'un model perfecte del sistema i pels seus requeriments i cost computacional. Tot i així, la programació dinàmica és conceptualment important, ja que és el que s'intenta imitar amb la majoria de mètodes de RL, però esquivant la necessitat del model i amb un cost computacional menor.

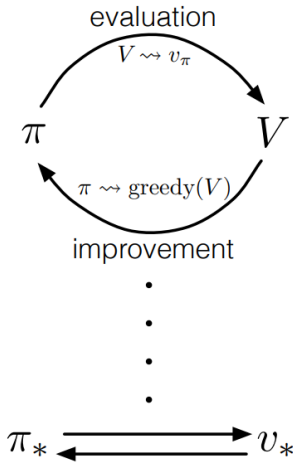

Figura 2: Procés iteratiu de la GPI. *Font:* [28]

La idea clau és l'ús de les *value functions* per organitzar i estructurar la cerca de polítiques òptimes. Com s'ha comentat, és fàcil determinar polítiques òptimes un cop s'ha determinat les funcions valor v_* o q_* que satisfan les equacions d'optimitat de Bellman (Equació 13 i Equació 14). Existeixen diversos mètodes i algorismes: *policy evaluation, policy improvement, policy iteration, value iteration*, etcètera.

El terme *generalized policy iteration* (GPI) fa referència a la idea que la *policy evaluation* i el *policy improvement* interactuïn (Figura 2). Quasi tots els mètodes de RL es descriuen com a GPIs, ja que consten de polítiques i funcions valor; les polítiques millorades respecte les *value funtions* i les *value functions* actualitzades segons al política. Aquests dos processos d'avaluació i millora s'estabilitzen quan la política trobada π és avariciosa respecte la seva funció valor v_π o q_π. Això significa que es compleix l'equació d'optimitat de Bellman.

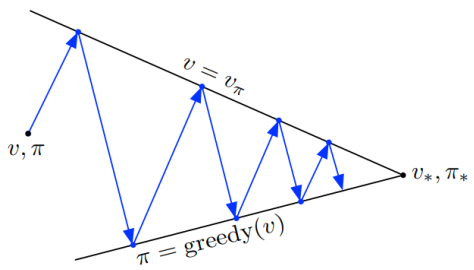

Figura 3: Idealització de la convergència de la GPI. *Font:* [28]

Es podria concebre aquesta interacció dels processos com una competició i a la vegada una cooperació. Fer la política *greedy* respecte la seva funció de valor normalment implica que la funció valor esdevingui incorrecta, i fer la funció valor consistent amb la política implica que aquesta deixi de ser *greedy*. Tot i així, a llarg termini, els dos processos estan cooperant per trobar una solució comuna: la funció valor òptima i la política òptima.

4.3.3 Mètodes de Monte Carlo (MC)

Aquests algorismes tabulars serveixen per aprendre la funció estat-valor $V(s)$ per a una política determinada. Recordem que la funció estat-valor d'un estat és el retorn esperat (recompensa acumulada amb descompte esperada) des d'aquest estat. Una manera intuïtiva d'estimar els valors seria fent la mitjana dels retorns observats després de visitar un estat. Com més retorns s'observin, més hauria de convergir la mitjana al valor esperat. Aquest és el principi en que es basen els mètodes de Monte Carlo. Aquests actualitzen els valors $v(s)$ un cop completat l'episodi i es coneix el retorn total des de cada estat. Per tant només és aplicable en *episodic tasks*. En cada episodi, el valor de cada estat s'actualitza fent la mitjana de tots els retorns vistos en els episodis anteriors i l'actual. Hi ha diferents tipus de MC. Per exemple, els *first-visit* només consideren el retorn de l'estat el primer cop que es visiten en un mateix episodi. En canvi els *every visit* tenen en compte totes les visites.

Els principals trets diferenciadors amb els models de programació dinàmica són dos. El primer, que no requereix d'un model ja que aprèn directament a partir de mostres de trajectòries d'episodis. El segon és que no fan servir *bootstrap*, és a dir, que no basen l'actualització dels valors de q o v en els valors d'altres estats.

4.3.4 Temporal-differnece Learning

Una de les idees centrals i innovadores de l'aprenentatge per reforç fou la introducció del *temporal-difference (TD) learning*. Aquest és una combinació dels mètodes de Monte Carlo i de programació dinàmica. Com els MC, els mètodes TD són tabulars i poden aprendre directament de l'experiència, és a dir, sense necessitat d'un model de l'entorn. I com els mètodes de DP, actualitzen les estimacions fent us d'altres estimacions sense esperar al retorn final de l'episodi (*bootstrap*). Per exemple,$v(s)$ fent ús de $v(s')$, on s' és l'estat futur. Per a un *every-visit* MC l'actualització es dona al final de l'episodi com:

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t)\right], \tag{15}$$

on α és una paràmetre constant que defineix el *step-size* de l'actualització. Per a un TD, l'actualització es pot donar al següent pas de temps. Un mètode simple TD realitza l'actualització a l'instant $t+1$

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right], \tag{16}$$

on el valor objectiu ha passat de ser G_t per al MC, a $R_{t+1} + \gamma V(S_{t+1})$ per al TD. De fet, aquest mètode TD s'anomena TD(0) o *one-step* TD, que és un cas especial dels mètodes TD(λ) i *n-step* TD (consultar [28]).

Finalment notar que l'expressió entre claudàtors de l' Equació 16 es pot entendre com un error entre el valor estimat de $V(S_t)$ i el valor objectiu estimat $R_{t+1} + \gamma V(S_{t+1})$. A aquesta quantitat

se l'anomena *TD error* :

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{17}$$

4.3.4.1 SARSA: On-policy TD

En comptes d'aprendre la funció estat-valor (V), aquest mètode estima la funció acció valor, és a dir, $q_\pi(s, a)$ per a la política π i per a tots els estats s i accions a. El seu nom és degut a la necessitat de $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$ per realitzar una actualització :

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_t + 1) - Q(S_t, A_t)\right] \tag{18}$$

La convergència de SARSA depèn de la naturalesa de la política. Si aquesta és una política avariciosa (greedy), s'enfronta al mateix problema que els MC: la no visita de moltes parelles estat-acció. Així doncs, polítiques habituals són les ε-$soft$, on $\pi(a \mid s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$ per a qualsevol s i a, o les ε-$greedy$, on la probabilitat d'escollir l'acció avariciosa és $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}$ i la de la resta d'accions és $\frac{\varepsilon}{|\mathcal{A}(s)|}$.

SARSA convergeix amb probabilitat 1 a una política i una funció acció-valor òptimes, sempre i quan es visitin tots els parells estat-acció un nombre infinit de vegades i la política convergeixi al límit en una política *greedy* (per exemple fent $\varepsilon = 1/t$ per a una política ε-$greedy$).

Finalment es mostra el pseudocodi de l'algorisme SARSA:

Algorisme 1 *SARSA (on-policy TD)* per estimar $Q \approx q_*$

Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize: $Q(s, a)\ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

for each episode **do**
  Initialize S
  Choose A from S using policy derived from Q (e.g, ε-$greedy$)
  while S is not terminal **do**
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g, ε-$greedy$)
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'; A \leftarrow A';$
  end while
end for

4.3.4.2 Q-learning: Off-policy TD

Un dels avenços més significatius en el món de l'aprenentatge per reforç és el desenvolupament d'aquest algorisme tabular, *off-policy* i *model-free* conegut com *Q-learning* (Watkins, 1989 [30]). Aquest és segurament el més conegut d'entre els mètodes TD, i aprèn i aproxima la funció acció-valor, independentment de la política seguida. Això simplifica substancialment l'anàlisi de l'algorisme i possibilita una convergència més ràpida. La política de comportament

seguida durant l'aprenentatge determina els estats i accions explorats. Per tant, per convergir, és necessari que aquesta política possibiliti l'actualització de tots els valors de totes les parelles estat-acció. Sovint es pren una política de comportament ε-$greedy$. Durant l'aprenentatge hi ha un balanç entre *exploitation* (acció avariciosa) i *exploration* (altres accions). En canvi, a l'hora d'actualitzar el valor de Q, es fa sempre amb la política objectiu avariciosa. Aquesta és al diferència que presenta respecte el SARSA, que actualitza el valor Q amb la mateixa política de comportament.

Al Q-$learning$ l'actualització dels valors es dona segons:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{19}$$

En aquesta podem identificar i desglossar diferents parts:

$$\underbrace{Q(s,a)}_{\text{new value}} \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \Bigg[\overbrace{\overbrace{\underbrace{r}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_{a'} Q(s',a')}_{\text{estimated optimal future value}}}^{\text{learned value}} - \underbrace{Q(s,a)}_{\text{old value}}}^{\text{TD-error}} \Bigg]$$

L'algorisme *Q-learning* es presenta en format de pseudocodi a continuació:

Algorisme 2 *Q-learning* (*off-policy TD*) per estimar $\pi \approx \pi_*$

Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize: $Q(s,a)\ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal,\cdot) = 0$

for for each episode **do**
  Initialize S
  while S not terminal **do**
    Choose A from S using policy derived from Q (e.g, ε-$greedy$)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha \left[R + \gamma \max_a Q(S',A') - Q(S,A) \right]$
    $S \leftarrow S'$
  end while
end for

Aquest és l'algorisme implementat en aquest projecte per donar solució al problema d'assignació de treballs en un entorn dinàmic.

5 Caracterització del problema d'estudi

En els apartats anteriors s'ha fet un repàs de les tècniques usades per a resoldre el DJSSP i també s'ha fet una introducció a l'aprenentatge per reforç. S'han descrit alguns algorismes inclòs l'usat per resoldre el problema. Abans de presentar la implementació feta i els resultats obtinguts, és necessari fer una caracterització del problema, establint les hipòtesis i condicions de contorn considerades. Com s'ha comentat, les variants del DJSSP són nombroses i, per tant, és necessària aquesta caracterització del problema.

5.1 Hipòtesis i condicions de contorn

Aquestes **hipòtesis i condicions de contorn** són:

1. N : nombre de treballs a realitzar. $(i = 1...N)$
2. M : nombre de màquines. $(j = 1...M)$
3. K : nombre de tipus d'operacions diferents. $(k = 1...K)$
4. J_OPS_i : Seqüencia d'operacions del treball *i*.
   Ex: $J\_OPS_i = [a, b, c]; \quad a, b, c \in [1, ..K]$
5. M_OPS_j : Conjunt d'operacions que pot processar la màquina *j*.
   Ex: $M\_OPS_j = \{a, b\}; \quad a, b \in [1, ..K]$
6. $T_{k,j}$: Temps de processament de l'operació de tipus *k* a la màquina *j*. Són coneguts amb certesa; no presenten aleatorietat. Nombre enter positiu $(u.t)$
7. TC : Temps de canvi entre tipus operacions en una màquina. Es considera igual per a totes les màquines i qualsevol parella de tipus d'operacions. Nombre enter positiu $(u.t)$
8. Cada màquina només pot processar una operació a la vegada.
9. Cada operació només pot ser iniciada si, en cas de tenir una operació precedent, aquesta ha estat finalitzada.
10. Cada operació de cada treball només es processa una vegada.
11. Una operació es comença i s'acaba en una mateixa màquina sense interrupcions.
12. No existeixen dependències entre treballs.
13. Els treballs no tenen cap mena d'índex de prioritat.
14. A l'inici, totes les màquines estan disponibles i preparades per a rebre qualsevol operació sense haver de realitzar un canvi de tipus d'operació.
15. Temps discretitzat en unitats de temps enteres $(u.t)$.

16. Temps de preparació i transport considerats nuls.

5.2 Esdeveniments estocàstics

Com s'ha vist a la Secció 3.3, la majoria d'implementacions consideren un sol esdeveniment estocàstic, en part, perquè com més tipus d'esdeveniments es considerin, més ampli és l'espai d'estats accessible. En la implementació que es proposa en aquest document s'han considerat dos **esdeveniments estocàstics** que conviuen en el mateix entorn:

1. Fallada d'una màquina o aturada per manteniment. Aquesta romandrà aturada un temps incert. No es contempla una fallada o aturada mentre la màquina està processant una operació; només quan estigui esperant o just en acabar de processar una operació.

2. Instants d'arribada dels treballs.

5.3 Objectius

D'altra banda, els **objectius** perseguits en aquest treball són:

1. Minimitzar el temps final d'acabament de l'últim treball (*makespan*).

2. Minimitzar el nombre de canvis d'operació. A més a més de suposar un cost temporal, també poden suposar un cost econòmic.

Si bé és cert que aquests dos objectius semblen anar alineats, la solució òptima d'un, no necessàriament ha de coincidir amb l'òptima de l'altre. A la Secció 6 es detallarà la funció de recompensa, que haurà d'estar alineada amb aquests objectius.

6 Implementació : *Q-learning*

En aquesta secció es descriu la solució implementada. Aquesta s'ha realitzat amb el mètode *Q-learning* (veure Secció 4.3.4.2). En primer lloc, es detalla la modelització del problema en el marc d'un MDP.

6.1 Defincició del MDP

Recordem que la modelització d'un problema de presa de decisions es formalitza mitjançant un *Markov Decision Proces*. Aquest queda totalment definit per la tuple d'elements (S, A, P, R, γ) (veure Secció 4.2). Així doncs, per al problema d'escrit a la Secció 5, cal definir cada un d'aquests elements. Ara bé, el model que defineix les probabilitats de transició (P) podria resultar difícil d'implementar donada l'aleatorietat dels instants d'arribada dels treballs i les fallades de les màquines. D'altra banda, l'algorisme Q-*learning* usat per a resoldre el DJSSP plantejat és un mètode TD que no requereix d'aquest model ja que es basa en l'experiència (*model-free*).

6.1.1 Espai d'estats

Per a que la implementació sigui correcta i es compleixi la Propietat de Markov (Equació 1), és necessari que la representació de l'estat inclogui la informació necessària de tots els aspectes que afecten el futur. La informació que s'ha considerat necessària és:

- Per a cada màquina *j*:
  - **Estat de la màquina** : $m\_status_j \in \{available, not\_available\}$
    Disponible (*available*) per a processar alguna operació o no disponible (*not available*) si ja està processant una operació, està espatllada o en manteniment.
  - **Última operació processada :** $last\_op_j \in \{1, ..., S, -1\}$
    Indica el tipus de l'última operació processada. A l'instant inicial i després d'una fallada o aturada de la màquina , aquest valor és -1, indicant que la màquina està preparada per a rebre qualsevol de les operacions que pot processar sense haver de realitzar l'operació de canvi.
- Per a cada treball *i*:
  - **Estat del treball** : $j\_status_i \in \{to\_arrive, waiting, processing, done\}$
    Els noms dels estats són autoexplicatoris.
  - **Pròxima operació a processar** : $next\_op_i \in \{1, ...S, -1\}$
    Quan el treball encara no ha arribat o ja s'ha processat aquest valor és -1; altrament indica la següent operació a realitzar d'aquest treball.

Així doncs, la tuple que descriu l'estat del sistema en termes generals seria:

$$state = (m\_status_1,\ last\_op_1, ...,\ m\_status_M,\ last\_op_M,\\ j\_status_1,\ next\_op_1, ...,\ j\_status_N,\ next\_op_N)$$

Un factor important és la dimensió d'aquest espai d'estats $|\mathcal{S}|$. Al ser un problema combinatori, l'espai d'estats pot esdevenir intractable (per alguns mètodes). Els possibles estats d'una màquina m_status_j són dos, i el nombre de valors que pot prendre $last\_op_j$ és igual al nombre d'operacions que pot processar la màquina j (que denotarem com $|M\_OPS_j|$) més 1. Sota la hipòtesi que cada un dels treballs és sempre el mateix en els diferents episodis (mateixes operacions), el nombre de possibles estats d'un job j_status_i són quatre, i el nombre de possibles valors de $next\_op_i$ és igual al nombre d'operacions de les que consta el treball (que denotarem com $|J\_OPS_i|$) més 1. La dimensió de l'espai d'estats es calcula com:

$$|\mathcal{S}| \quad = \quad \prod_{j=1}^{M} \left[\left(2 \left(|M\_OPS_j| + 1\right)\right] \quad \cdot \quad \prod_{i=1}^{N} \left[4 \left(|J\_OPS_i| + 1\right)\right] \tag{20}$$

Quan el nombre d'operacions a realitzar de cada job i el nombre d'operacions que pot processar cada màquina és el mateix, es pot reformular com:

$$|\mathcal{S}| \quad = \quad \left[2 \left(|M\_OPS_j| + 1\right)\right]^{M} \quad \cdot \quad \left[4 \left(|J\_OPS_i| + 1\right)\right]^{N} \tag{21}$$

D'aquesta manera es comprova que l'espai d'estats creix exponencialment en augmentar el nombre de treballs i/o de màquines. Per exemple, suposem que $|M\_OPS_j| = 3$ i $|J\_OPS_i| = 3$ i M = 2. Vegem l'evolució de $|\mathcal{S}|$ en funció de N:

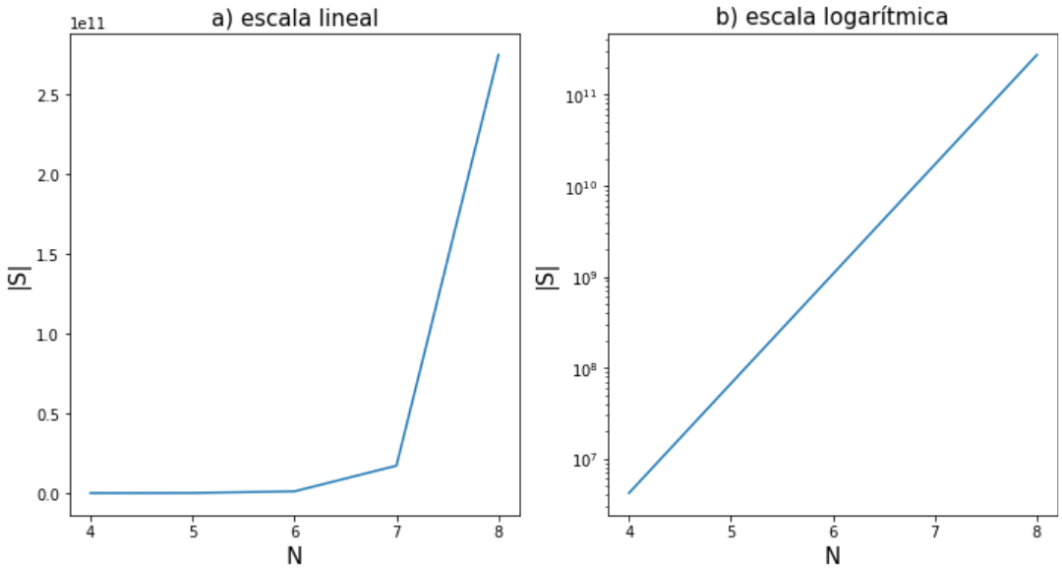

Figura 4: Creixement de l'espai d'estats en augmentar N, per M=2 i |J_OPS| = |M_OPS| = 3.
Font: Elaboració pròpia

Si, en canvi, considerem que els treballs poden ser de qualsevol tipus, l'expressió de l' Equació 20 esdevé:

$$|\mathcal{S}| \quad = \quad \prod_{j=1}^{M} \left[\left(2 \left(\left| M\_OPS_j \right| + 1 \right) \right] \right. \quad \cdot \quad \prod_{i=1}^{N} \left[4 \left(K + 1 \right) \right] \tag{22}$$

on K és el nombre de tipus d'operacions diferents.

Si suposem un entorn més desafiant, mantenint nombre d'operacions de cada màquina i de cada treball igual a 3 però amb 20 treballs i 4 màquines, el nombre d'estats possibles és de $|\mathcal{S}| = 4,95 \cdot 10^{27}$ segons l' Equació 20 i de $|\mathcal{S}| = 1,64 \cdot 10^{31}$ segons l' Equació 22.

Aquestes dimensions no serien tractables amb un mètode tabular. Tot i així, aquest càlcul s'allunya força de la realitat per diversos motius. Alguns d'ells:

- El nombre de treballs que s'estan processant en el mateix instant de temps serà menor o igual al nombre de màquines no disponibles.
- Per a cada treball, no existeixen les combinacions on l'estat és *waiting* i l'operació següent a processar -1, és a dir, cap.
- En la mateixa línia, tampoc es poden tenir treballs en estat *to arrive* o *done* amb la següent operació a processar diferent de -1.
- Tot i que l'instant d'arribada dels treballs presenta certa aleatorietat, aquests arriben en un cert ordre. No es tindran configuracions on els darrers treballs estiguin processant-se o esperant i els primers encara hagin d'arribar. Serà difícil també tenir combinacions on els darrers treballs estan acabats i els primers encara s'estan processant.
- Donada la probabilitat de fallada de les màquines, és difícil tenir més maquines aturades per fallada o manteniment que disponibles o ocupades processant alguna operació.

Per aquests motius i d'altres, es fa pràcticament impossible determinar amb exactitud la dimensió de l'espai d'estats. A l'hora de simular diversos entorns, s'ha vist que el nombre d'estats realment accessible és molt menor del que es podria esperar amb l' Equació 20 o l' Equació 22.

6.1.2 Espai d'accions

Les accions que l'agent pot escollir són de dos tipus: **assignació** o **esperar**. L'acció d'esperar és present en qualsevol estat. Les d'assignació són diverses, no sempre estan disponibles i es representen com:

$$action = (job,\ operation,\ machine)$$

Aquesta tuple indica quina operació s'assigna a quina màquina i a quin treball correspon aquesta operació. És necessari puntualitzar que cada iteració no produeix necessàriament un increment de temps en el sistema. És important no confondre el número d'iteració, normalment definit com t, amb l'instant de temps del sistema real. Per evitar confusions, en aquest projecte

es denomina aquest "rellotge"del sistema amb la variable *time*. La raó d'aquesta diferenciació és simple: si és possible realitzar més d'una assignació en un instant de temps de la producció determinat, no tindria sentit fer una sola assignació i avançar el temps. Molt sovint es tindrien màquines amb temps morts i el temps de producció total es veuria afectat. Així doncs, la solució que es proposa és que les accions de tipus assignació no vagin mai succeïdes per un increment en el temps. Només l'acció d'esperar anirà succeïda, sempre, per un augment del temps. Donat que esperar no produeix cap modificació en l'estat, és necessari un increment del temps per possibilitar l'arribada d'un nou treball o que una màquina passi a estar disponible i es modifiqui així l'estat. D'aquesta manera s'eviten també bucles infinits quan l'agent ja ha acabat l'entrenament.

Suposem un exemple senzill, on un treball que consta de dues operacions (1 i 2) s'ha de processar en una sola màquina. La màquina necessita 2 u.t per processar l'operació 1 i 1 u.t per processar l'operació 2. La seqüència d'estats i accions en cada iteració seria la següent:

t	S_t	A_t	$time_t$
0	$(av, -1, wait, 1)$	$(J, 1, M)$	0
1	$(not\ av, 1, proc., 2)$	$rest$	0
2	$(not\ av, 1, proc., 2)$	$rest$	1
3	$(av, 1, proc., 2)$	$(J, 2, M)$	2
4	$(not\ av, 2, proc., -1)$	$rest$	2
5 = T	$(av, 2, done, -1)$	-	3

Taula 2: Estats, transicions i variable *time* d'un exemple senzill del funcionament de l'entorn. *Font: Elaboració pròpia*

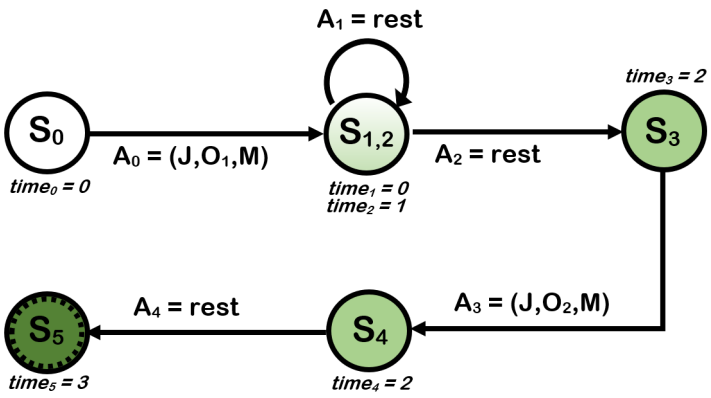

Figura 5: Diagrama de transicions d'un exemple senzill del funcionament de l'entorn *Font: Elaboració pròpia*

Amb el petit exemple anterior s'exemplifica el funcionament i les transicions que es comentaven. És important observar que la variable $time_t$ augmenta en una unitat de temps només quan l'acció $A_{t-1} = rest$. Una acció $A_t = rest$ pot conduir a un estat S_{t+1} igual o diferent a S_t.

Les accions d'assignació sempre modifiquen l'estat però mai la variable del temps del sistema $(time)$.

De manera anàloga al que s'ha fet amb l'espai d'estats, es proposa un càlcul de la dimensió de l'espai d'accions, és a dir, totes les possibles combinacions per a la tuple $(job, operation, action)$ més l'acció d'esperar.

$$|\mathcal{A}| \quad = \quad \left(M \cdot \sum_{i=1}^{N} |J\_OPS_i| \right) + 1 \tag{23}$$

Altre cop, aquest càlcul es realitza sota la hipòtesi que cada un dels treballs constarà de les mateixes operacions en tots els episodis. En cas de no ser així, aquesta dimensió augmenta i passa a ser:

$$|\mathcal{A}| \quad = \quad (M \cdot K) + 1 \tag{24}$$

on K és el nombre de tipus d'operacions diferents que hi ha.

Ara bé, el nombre d'accions possibles en un estat $\left|A(s)\right|$ és menor a l'espai d'accions total $|A|$. La raó és que moltes accions d'assignació $(job, operation, machine)$ no són possibles en funció de l'estat. A continuació es mostren les diferents condicions que impossibiliten una assignació:

- $m\_status_{machine} = not\ available$
- $j\_status_{job} = to\ arrive$
- $j\_status_{job} = processing$
- $operation \neq next\_op_{job}$
- $operation \notin M\_OPS_{machine}$

L'acció d'esperar és possible en qualsevol estat. Una opció és fer que l'agent aprengui també a distingir les accions prohibides, donant-li una forta penalització quan n'esculli una. Però com que $\left|A(s)\right| \ll |A|$, l'entrenament resulta ser molt més llarg, ja que per a cada estat ha de provar moltes més accions. En aquest problema és fàcil acotar l'espai d'exploració de l'agent només a les accions possibles en cada estat i, d'aquesta manera, reduir el temps d'entrenament substancialment (aproximadament un 75-80%). Per tant, en cada estat, a l'agent només se'l deixa escollir accions possibles.

6.1.3 Funció de recompensa

El senyal de recompensa que l'agent rep en prendre una acció en un estat determinat és la base de l'aprenentatge. Sense recompenses no es pot construir una política òptima. Aquesta recompensa serveix per informar a l'agent de l'objectiu que es persegueix. Recordem que els objectius plantejats en aquest projecte són dos:

1. Minimitzar el temps de finalització de tots els treballs.

2. Minimitzar el nombre de canvis d'operació.

Per al primer objectiu, una implementació usual és penalitzar amb una recompensa negativa (per exemple de -1) qualsevol acció que no condueixi a un estat terminal i una recompensa de +1 quan s'assoleix un estat terminal. D'aquesta manera, l'agent, en el seu intent de maximitzar el retorn al llarg de l'episodi, provarà de fer el mínim d'accions possibles. Com més accions faci, més penalitzacions de -1 rebrà. Aquesta és l'estratègia que s'ha seguit, amb alguns matisos.

Com que cada cop que s'executa l'acció d'**esperar**, seguidament es produeix un increment del temps de rellotge del sistema (*time*), s'ha decidit que si $a = rest$, la $reward = -1$, excepte quan l'estat següent sigui terminal. En aquest cas, esperar serà l'única opció i, per tant, podem deixar $reward = 0$. En aquests estats previs a l'estat terminal, l'acció d'esperar serà l' única acció vàlida i, per tant, serà l'única acció amb un valor de $q(s, a) = 0$. Un valor de 0, és el màxim possible de tota la *Q_table*; les accions tendiran a dirigir-se cap aquest estat per tal de no seguir rebent recompenses negatives.

D'altra banda, a les accions de tipus **assignació** se'ls dona una recompensa de 0. Una assignació anirà succeïda ,de manera inevitable, d'un nombre determinat d'accions futures del tipus espera, ja que farà que la màquina on s'ha fet l'assignació estigui ocupada durant un cert període i, per tant, no estigui disponible per processar cap més operació. És per aquest motiu que l'agent, de manera indirecte, provarà de prioritzar aquelles assignacions que impliquin una no disponibilitat d'una màquina menor. Ara bé, perseguint el segon objectiu, d'evitar canvis d'operacions pel cost temporal i econòmic que poden suposar, s'estableix una recompensa de -1 per aquelles assignacions que impliquin un canvi d'operació en una màquina.

Es poden distingir doncs dos senyals de recompensa. El primer (R_1), aniria alineat amb l'objectiu de reduir el temps de finalització de tots els treballs i aplica només quan l'acció és esperar. El segon (R_2), només s'aplica quan l'acció és una assignació i penalitza els canvis d'operació.

$$R_1 = \begin{cases} 0, & \text{if } a = rest \text{ and } s' \textbf{ is} \text{ terminal} \\ -1, & \text{if } a = rest \text{ and } s' \textbf{ is not} \text{ terminal} \end{cases} \tag{25}$$

$$R_2 = \begin{cases} 0, & \text{if } a \neq rest \text{ and } operation = last\_op_{machine} \\ -1, & \text{if } a \neq rest \text{ and } operation \neq last\_op_{machine} \end{cases} \tag{26}$$

A partir d'aquests dos senyals de recompensa es pot generar una recompensa global com a suma ponderada d'aquestes dues. L'usuari pot definir quina importància vol donar a cada objectiu amb uns pesos w_1 i w_2:

$$reward = \frac{w_1}{w_1 + w_2} \cdot R_1 + \frac{w_2}{w_1 + w_2} \cdot R_2 \tag{27}$$

A continuació es presenta el pseudocodi de la funció de recompensa:

Algorisme 3 Funció de recompensa

Parameters : action a, actual state s, future state s', weights w_1, w_2
Initialize: $R_1, R_2 = 0$
if $a = rest$ **then**
  if $s' is\ terminal$ **then**
    $R_1 = 0$
  else
    $R_1 = -1$
  end if
else
  $job, operation, machine \leftarrow a$
  if $operation \neq last\_op_{machine}$ **then**
    $R_2 = -1$
  else
    $R_2 = 0$
  end if
end if
$reward \leftarrow \frac{w_1}{w_1+w_2} \cdot R_1 + \frac{w_2}{w_1+w_2} \cdot R_2$
Return : $reward$

6.1.4 Factor de descompte

Els valors de $Q(s, a)$ s'actualitzen a cada iteració segons l'expressió de l' Equació 19, on és necessari definir el factor de descompte per a les recompenses futures. Recordem que com més pròxim a 0 és aquest valor, més importància es dona a la recompensa immediata i menys a les futures. Per al nostre problema, i donada la naturalesa del senyal de recompensa, les recompenses futures són igual o més importants que les immediates. Per aquest motiu es decideix que $\gamma = 1$.

6.2 Implementació

La implementació s'ha realitzat en llenguatge de programació *Python*. Tot i l'existència de llibreries, com *gym* de OpenAI ([31]), que faciliten la creació d'un entorn, s'ha decidit modelitzar el sistema des de zero, fent ús de les llibreries *built-in*, *Pandas* i *NumPy*. El codi es pot consultar lliurament al repositori GitLab ([34]).

Com s'ha explicat anteriorment, se sospita que l'espai d'estats realment accessible és molt menor al nombre d'estats possibles segons els càlculs explicats (Equació 20 o Equació 22). Normalment, a l'inici de l'algorisme s'inicialitza una matriu amb tantes files com estats i tantes

columnes com accions. Ara bé, per al cas que es proposava a la Secció 6.1.1, el nombre d'estats era de l'ordre de 10^{27}. Independentment del nombre d'accions, aquesta matriu no és tractable i no té sentit inicialitzar-la tota si la majoria dels estats no són accessibles. Una alternativa és representar la matriu en forma de diccionari, en concret, un *defaultdict*. Cada clau del diccionari serà un estat i els valors seran *np.arrays* dels *q-values* de les diferents accions. Aquest tipus de diccionari s'inicialitza buit i cada cop que s'intenta accedir a un estat (clau) no existent, no retorna un error sinó el que l'usuari desitja. En el cas que ens ocupa, si no s'ha visitat mai aquell estat prèviament, el *defaultdict* retorna un *np.array* de zeros. El motiu pel qual s'inicialitza amb zeros és que un *q-value* de 0, és el màxim possible, ja que totes les recompenses són zero o negatives. D'aquesta manera es potencia l'exploració encara que es segueixi un comportament *greedy*. El paràmetre èpsilon ε de la política de comportament ε-$greedy$ permet definir el balanç *exploration-exploitation*. Tot i així, es pot assegurar la convergència del *Q-learning* per a qualsevol política de comportament. Per tal de veure l'evolució del retorn, s'usa una política de comportament ε-$greedy$, on ε_0 és 1 al primer episodi, i en cada episodi es redueix tal que $\varepsilon_{n+1} = \varepsilon_n - \varepsilon_0/num\ episodis$, on n és el número d'episodi. D'aquesta manera, sigui quin sigui el nombre d'episodis entrenats, la política de comportament convergeix a la política objectiu al final de l'entrenament. La literatura presenta altres maneres de variar aquesta ε al llarg dels episodis. Per exemple, sovint es proposa que $\varepsilon(n) = M^n$ on n és el número d'episodi i M un valor proper a 1. D'aquesta manera s'aconsegueix una aproximació d'èpsilon asimptòtica a zero. Si es vol evitar una disminució massa brusca i precoç o una disminució massa lenta i insuficient, aquest valor M hauria de dependre del nombre d'episodis totals.

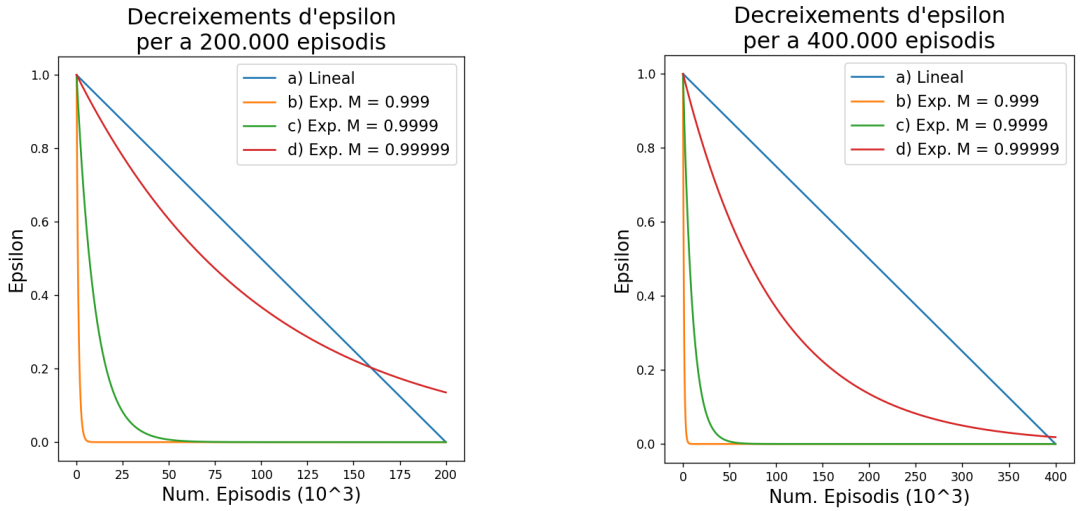

Figura 6: Decreixements de ε lineals i exponencials per $2 \cdot 10^5$ i $4 \cdot 10^5$ episodis. *Font: Elaboració pròpia*

Un decreixement lineal és la manera més senzilla i generalitzable. Aquest decreixement implica que, a mesura que l'entrenament avança, l'exploració es va reduint i l'explotació (comportament avariciós) va augmentant. D'aquesta manera, l'aleatorietat en la tria de decisions es va reduint

i, per tant, el retorn hauria d'anar augmentant.

Ara bé, la inicialització del vector de valors de cada estat $Q(s)$ inclou valors de moltes accions prohibides. Aquests valors no es modificaran mai ,ja que, com s'ha dit, no es dona la possibilitat a l'agent d'escollir-les. Per tant, la inicialització serà $Q(s,a) = L$ quan $a \notin A(s)$ i $Q(s,a) = 0$ quan $a \in A(s)$. Aquest valor de L no es modificarà mai i, per consegüent, ha de ser un valor negatiu i de valor absolut suficientment gran, tal que un cop realitzat l'entrenament $Q(s, a \notin A(s)) < Q(s, a \in A(s)) \ \forall s, a$. És a dir, L ha de ser suficientment petit com per ser sempre la pitjor opció i que l'agent, amb una política *greedy*, no esculli mai aquesta acció prohibida. Depenent de la dimensió del problema, el nombre de passos per arribar a l'estat terminal serà major o menor. Si augmenta la dimensió del problema, els retorns esperats augmenten en valor absolut. Donades les dimensions dels entorns que es tracten, un valor de $L = -1000$ és suficient.

A l'hora de generar els entorns s'ha preparat un full de càlcul per tal d'introduir les dades de l'entorn: nombre de màquines (M), nombre de treballs (N), operacions de cada treball (J_OPS_i), operacions que pot realitzar cada màquina (M_OPS_j) temps de processament per a cada parella màquina-tipus d'operació $(T_{k,j})$ i el temps de canvi de cada operació (TC). També es proporcionen els instants de temps en que els treballs estan disponibles per començar a ser processats. Aquests però, s'alteren en cada episodi (dins un cert rang) per tal de realitzar l'entrenament en l'entorn estocàstic plantejat.

Pel que fa a la fallada de les màquines, s'estipula una probabilitat de fallada a l'hora d'entrenar. Aquesta correspon a la probabilitat que una màquina passi a tenir una fallada just en acabar de processar un treball (fallada de la màquina) o mentre està disponible sense processar cap operació (manteniment).

Finalment, comentar que per a la visualització de la planificació de treballs s'usa una llibreria de *Python* anomenada *plotly.express*, que conté un mòdul que és *timeline*([32]) . Aquest permet fer gràfics de tipus diagrama de Gantt que es poden adaptar per tal de visualitzar la seqüenciació d'operacions del problema tractat.

6.3 Simulacions i resultats

En aquest apartat es mostra l'entrenament i resultats per a tres entorns diferents. En tots ells, el temps de canvi es considera d'una unitat de temps $(TC = 1u.t)$ i la probabilitat de fallada d'una màquina del 10%; tot i que en un entorn real seria menor. En un primer entorn senzill (N=4, M=2), s'han analitzat diferents configuracions del paràmetre alfa i dels pesos de la funció de recompensa $(w_1$ i $w_2)$. Donada la variabilitat dels instants d'arribada dels treballs i les fallades de màquina, es fa impossible comparar els resultats amb diferents configuracions si no és eliminant aquests esdeveniments estocàstics a l'hora de simular una planificació. Per tant, tot i que l'entrenament es realitza en un entorn dinàmic, la simulació i els resultats presentats corresponen a l'entorn estàtic. Tot i així, se'n presenta una simulació per a una de les configu-

racions en condicions dinàmiques. Els altres dos entorns (N=6,M=3 i N=8,M=4) se simulen també en la seva vessant dinàmica. Tota la informació sobre els entrenaments realitzats es pot consultar de forma visual a l' Annex A , així com també els diversos exemples de simulació a l' Annex B. Per a les mètriques *Score*, temps de finalització i nombre de canvis d'operació, s'aplica una mitjana mòbil per tal de clarificar els gràfics.

Per a entorns més complexos, l'entrenament amb Q-*learning* ja ha sigut inviable. Per a un problema més complex (N = 15 i M = 4), després de 400.000 episodis s'havien visitat 97 milions d'estats i ha sorgit un error relacionat amb la memòria disponible. Això posa de manifest la limitació dels sistemes tabulars i la necessitat d'implementar mètodes aproximats si es vol abordar el problema en un entorn més complex ($|S| \cdot |A|$ major).

6.3.1 Entorn 1: N = 4 treballs i M = 2 màquines

Les característiques d'aquest primer entorn són les següents:

Treball Id	Operacions 1a	2a	3a	Arribada (u.t)
1	2	3	4	0
2	1	2	3	0
3	5	2	1	0
4	3	4	5	7

Taula 3: Operacions i arribades dels treballs de l'Entorn 1. *Font: Elaboració pròpia*

Màquina Id	Temps de processament (u.t) Op.1	Op.2	Op.3	Op.4	Op.5
1	2	2	3	3	3
2	3	3	3	2	2

Taula 4: Temps de processament de les màquines de l'Entorn 1. *Font: Elaboració pròpia*

Si es consideren entorns dinàmics, es fa molt difícil determinar una cota, com es fa quan es desenvolupen heurístiques per determinar-ne la seva qualitat. Una cota és la solució òptima del problema relaxant les condicions de contorn. En aquest cas, si eliminem els esdeveniments estocàstics i la seqüencialitat i indivisibilitat de les operacions, es podria determinar una cota inferior per al nombre de canvis i temps de finalització dels treballs. La cota inferior per al nombre de canvis és força intuïtiva. Donat que les màquines a l'inici de la producció estan preparades per processar qualsevol operació, no és necessari realitzar cap canvi. Així doncs, es podrien processar M tipus d'operacions sense haver realitzat cap canvi. El nombre de tipus d'operacions restants serà igual al nombre mínim de canvis.

$$C_{min}\ op\_changes = K - M \tag{28}$$

La cota inferior per al temps de processament final es calcula suposant que totes les operacions es processen en el menor temps possible, és a dir, a la màquina que la pot processar més ràpid. Aquest temps total de processament es divideix entre el nombre de màquines del sistema per obtenir els temps de processament de cada línia, suposant divisibilitat de les operacions i un equilibrat perfecte entre màquines. Seguidament, s'afegeix a cada línia el temps mínim que

dedicarà a canvis d'operació.

$$C_{min}\ makespan = \left\lceil \frac{\sum_{i=1}^{N} \sum_{op \in J\_OPS_i} \min_M T_{M,op}}{M} + \left\lceil \frac{K-M}{M} \right\rceil \cdot TC \right\rceil \tag{29}$$

Aquestes cotes no són representatives per al problema dinàmic (DJSSP), però sí per al problema estàtic que es considera per analitzar diferents configuracions d'entrenament. Així doncs, per al problema descrit a la Taula 3 i Taula 4, les cotes serien:

$$C_{min}\ op\_changes = 5 - 2 = 3 \text{ canvis}$$
$$C_{min}\ makespan = \left\lceil \frac{27}{2} + \left\lceil \frac{3}{2} \right\rceil \cdot TC \right\rceil = 16 \text{ u.t}$$

És necessari recordar que les cotes no representen la solució òptima del problema que es tracta. Depenent de la qualitat d'aquesta cota, pot estar més propera o menys a la solució del problema no relaxat.

Alfa és un hiperparàmetre que acostuma a prendre un valor de 0,2, però s'ha volgut també analitzar l'entrenament i els resultats per a un valor de 0,3, i veure si s'observen diferències.

D'altra banda, els pesos de la funció de recompensa, per donar més importància al temps de finalització o al nombre de canvis d'operació, són valors que l'usuari defineix segons la seva voluntat. S'ha entrenat el model per veure si amb uns pesos extrems s'aconsegueix realment minimitzar el temps de finalització o els canvis d'operació. També amb pesos intermedis iguals per a cada un dels objectius. La informació es recull a la següent Taula 5.

Cas	Alpha	w_1	w_2	Makespan	Op. changes	Visited states	Training time[2]
1	0.2	0.9	0.1	18	6	20 850	1:04:34
2	0.3	0.9	0.1	19	8	20 843	1:05:52
3	0.2	0.5	0.5	18	6	20 849	1:06:58
4	0.3	0.5	0.5	18	6	20 847	1:06:05
5	0.2	0.1	0.9	20	4	20 850	1:07:55
6	0.3	0.1	0.9	20	4	20 849	1:06:37

Taula 5: Resultats obtinguts per a diferents entrenaments de 2,5 milions d'episodis per a l'Entorn 1 estàtic. *Font: elaboració pròpia*

En primer lloc notar que el valor mínim de temps de finalització és 18 u.t i el valor mínim de

[2]Característiques del dispositiu utilitzat:AMD Ryzen 7 3700U amb targeta gràfica integrada Radeon Vega Mobile Gfx 2.30 GHz i RAM de 16,0 GB (utilitzable: 13,9 GB)

canvis d'operació és 4. Es comprova que, quan es dona un pes major a les penalitzacions per increment de temps (w_1=0.9), el temps de producció total és el més baix. En canvi, si es penalitza més els canvis d'operació, el nombre de canvis disminueix. Per tant, la funció de recompensa compleix el seu objectiu. Tot i així, per a valors intermedis (w_1=w_2=0.5) el temps de finalització també és el més baix trobat. Tot i que aquests valors de 18 u.t i 4 canvis no siguin iguals a les cotes inferiors abans descrites, no implica que no siguin solucions òptimes.

D'altra banda, el nombre d'estats visitats són vora els 20 850. Això representa aproximadament un 0,2 % del nombre de possibles estats calculats amb l'Equació 20. Com se sospitava, la dimensió de l'espai d'estats és molt inferior del que es podia esperar. En aquests mètodes tabulars, no existeix el que es coneix com a *overfitting* en l'aprenentatge supervisat. Si l'entrenament és massa extens, l'únic que passa és que s'està invertint temps en intentar millorar una política que ja ha convergit.

Primerament s'havia realitzat un entrenament d'un milió d'episodis. El nombre d'estats visitats no superava els 20 300. Analitzant la corba de l'evolució d'estats visitats, s'observa que aquesta no s'estabilitzava (Figura 7). Tal com s'ha definit la caiguda de ε, sempre s'aconseguirà un comportament tal que als últims episodis el nombre d'estats visitats es mantingui estable. Això és degut a que en els últims episodis es segueix una política pràcticament *greedy* i només es visiten estats ja visitats.

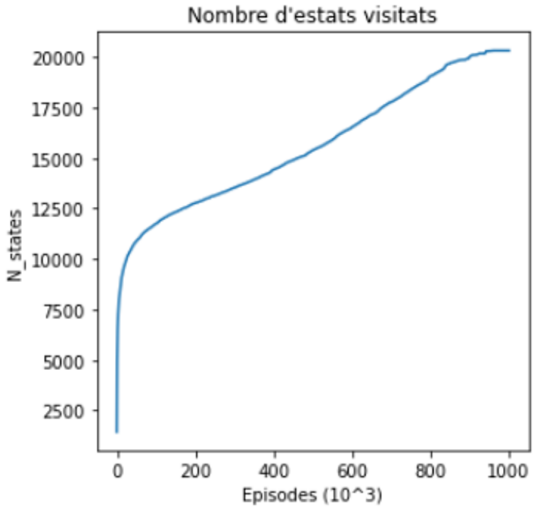

Figura 7: Exemple de corba d'estats visitats en 1 milió d'episodis per al Cas 3. *Font: Elaboració pròpia*

Per tant, la idea és arribar al màxim d'estats visitats quan èpsilon encara té un valor suficientment gran. És per això que s'ha realitzat un entrenament més extens, de dos milions i mig d'episodis, per veure si, efectivament, el sistema assolia una estabilitat.

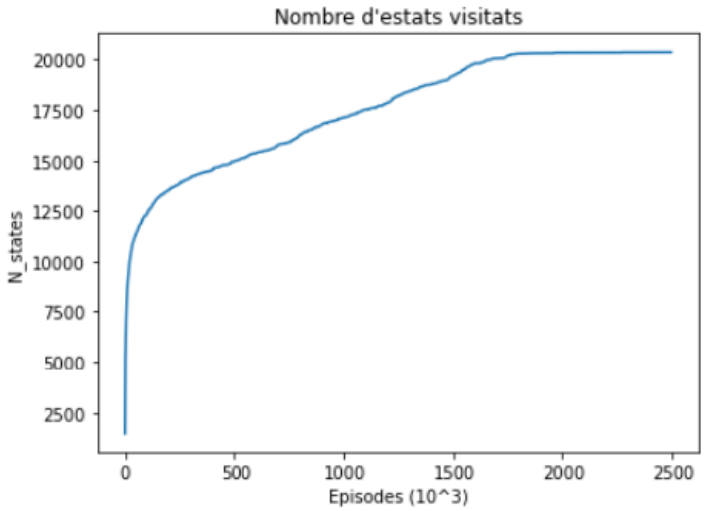

Figura 8: Exemple de corba d'estats visitats en 2,5 milions d'episodis per al Cas 3. *Font: Elaboració pròpia*

En aquest segon entrenament, s'observa que el nombre d'estats visitats arriba a un màxim quan el valor d'èpsilon és encara de 0,25. A més a més, no només es tracta de visitar tots els estats possibles, sinó de visitar-los el màxim nombre de vegades possible. Com més vegades es visiti cada estat i cada acció, més ajustats seran els valors $q(s, a)$ i més s'assegura és la convergència.

Tot i que els resultats de les simulacions en l'entorn estàtic han estat els mateixos per als dos entrenaments de 1 i 2,5 milions d'episodis, el segon és molt més fiable i és el que s'ha considerat.

Si s'analitzen els resultats obtinguts per als diferents ratis d'actualització (α) dels valors de la *Q-table*, no s'aprecien diferencies excepte per al cas $w_1 = 0.9$ i $w_2 = 0.1$. En aquest cas, el *makespan* i el nombre de canvis d'operacions és major en el cas $\alpha = 0.3$ que per $\alpha = 0.2$. En observar les mètriques d'entrenament a l' Secció A.1 (Figura 19, Figura 20), s'aprecia que per al mateix nombre d'episodis, per $\alpha = 0.3$ el *makespan* i el nombre de canvis comença a disminuir abans però $\alpha = 0.2$ acaba obtenint una millor solució.

Primerament es presenta una simulació on l'agent no ha estat entrenat i, per tant, pren accions de manera totalment aleatòria:

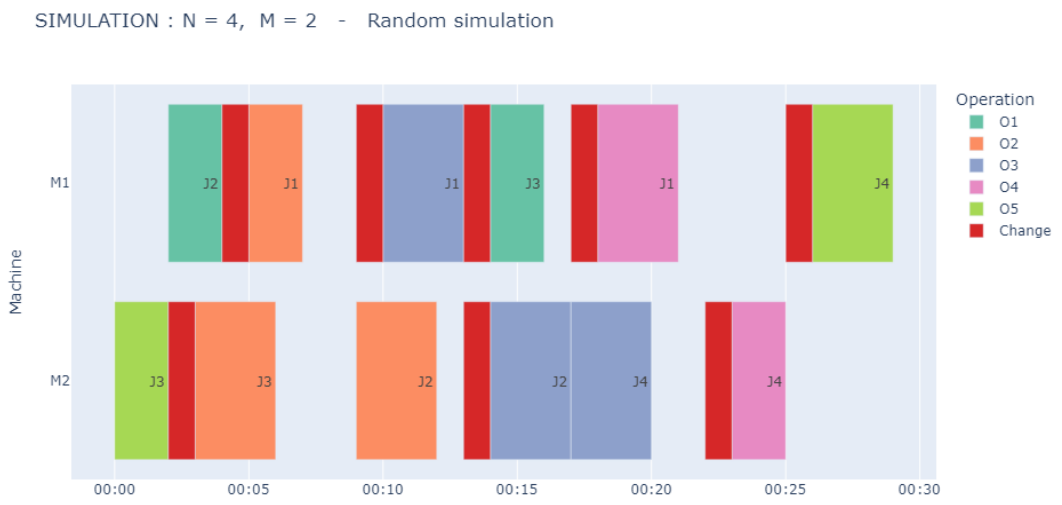

Figura 9: Simulació sense cap entrenament de l'agent de l'Entorn 1. *Font: Elaboració pròpia*

S'observa que la política de comportament no té cap mena de sentit. Es prenen decisions aleatòries (però factibles) en cada un dels estats. La solució presenta un temps de finalització de 30 unitats de temps, 8 canvis d'operació i multitud de temps morts; accions d'esperar quan hi ha possibles assignacions a fer. A mode d'exemple es presenten dues simulacions de l'entorn estocàstic per als Casos 1 i 5 de la Taula 5 un cop realitzat l'entrenament. A l' Annex B es poden consultar les simulacions dels altres casos.

Figura 10: Simulació de planificació del Cas 1 (Taula 5). *Font: Elaboració pròpia*

Figura 11: Simulació de planificació del Cas 5 (Taula 5). *Font: Elaboració pròpia*

Com es pot apreciar segons la llegenda, les franges vermelles indiquen un canvi d'operació. Es comprova que en minimitzar els canvis, augmenten els temps morts no productius deguts a precedències i això augmenta el temps de finalització de les tasques.

Per acabar, es mostra un exemple de planificació incorporant els esdeveniments estocàstics. Els tres primers treballs disponibles des de l'instant inicial i el quart treball a partir de l'instant 6 u.t. La màquina M2 pateix una aturada a l'instat 6 u.t d'una durada de 3 u.t

Figura 12: Simulació de planificació de l'entorn dinàmic del Cas 3 (Taula 5). *Font: Elaboració pròpia*

6.3.2 Entorn 2: N = 6 treballs i M = 3 màquines

En aquests cas, les arribades dels treballs no es presenten com un valor fix, sinó un rang de valors enters. Quan una màquina pateix una aturada per fallada o manteniment es manté en estat *no disponible* entre 1 i 4 unitats de temps. Es prenen uns pesos de $w_1 = w_2 = 0.5$, equiparant la importància dels dos objectius, i es manté $TC = 1\ u.t$

Treball	Operacions			Arribada
Id	1a	2a	3a	(u.t)
1	2	3	4	0
2	1	2	3	0
3	5	2	1	0
4	3	4	5	5-9
5	1	2	3	7-11
6	3	4	5	9-13

Taula 6: Operacions i arribades dels treballs de l'Entorn 2.

Màquina	Temps de processament (u.t)				
Id	Op.1	Op.2	Op.3	Op.4	Op.5
1	4	3	2	2	2
2	4	2	2	2	2
3	3	2	4	3	4

Taula 7: Temps de processament de les màquines de l'Entorn 2.

Un cop realitzat l'entrenament de 4 milions d'episodis, les mètriques i els resultats obtinguts són els següents:

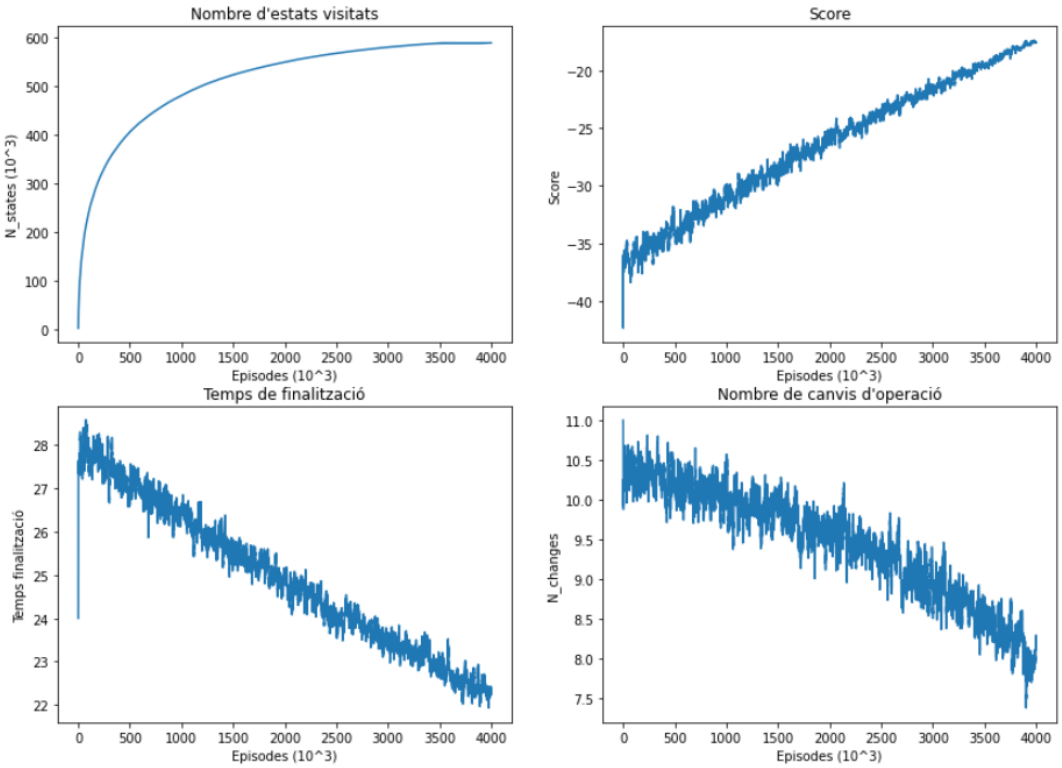

Figura 13: Mètriques de l'entrenament de l'Entorn 2 per a 4 milions d'episodis. *Font: Elaboració pròpia*

S'observa que, com és esperable, es redueixen els temps de finalització i el nombre de canvis a mesura que augmenta el nombre d'episodis realitzats. El nombre d'estats visitats creix de manera important als primers episodis i seguidament el pendent d'aquesta corba va tendint a 0. Per a aquest cas, la dimensió de l'espai d'estats segons Equació 20 seria de $2,8991 \cdot 10^{10}$, però el nombre d'estats visitats ha estat de 593 449, és a dir, un 0,002 %. El fet que la corba d'estats es vagi estabilitzant i assoleixi un pendent nul partir de l'episodi $3,6 \cdot 10^{6}$ (el nombre d'estats no augmenta) pot ser degut al decreixement d'èpsilon, tot i que, analitzant la corba, sembla que arribi a un màxim. S'ha comprovat que a partir dels 2,9 milions d'episodis, la solució per al problema estàtic no varia. Ara bé, no es pot comprovar que també ho sigui per a qualsevol cas en condicions dinàmiques. És per això que l'estabilitat assolida amb aquest entrenament no és tan clara com en l'Entorn 1.

Amb aquest entrenament es proposa una simulació de planificació de les operacions en condicions estocàstiques.

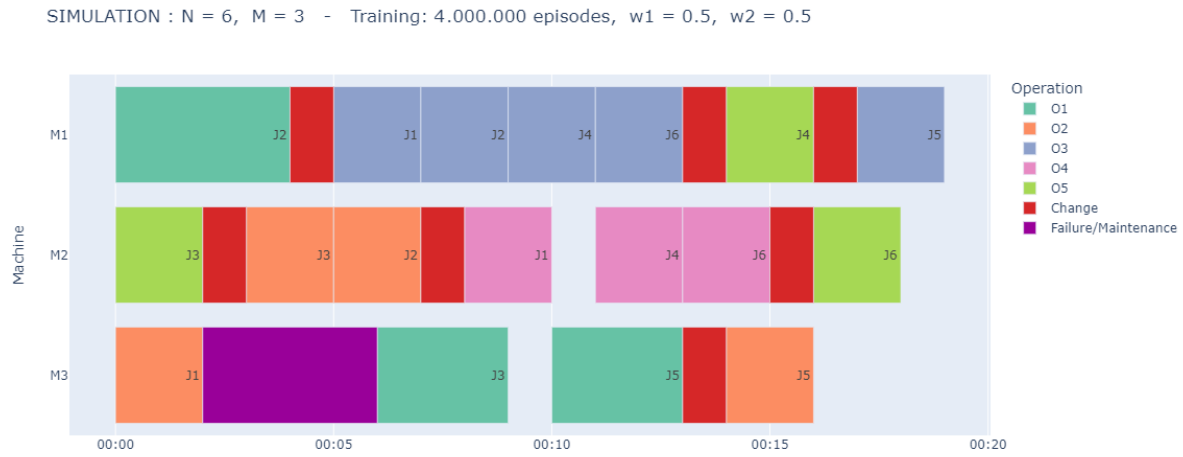

Figura 14: Simulació de planificació de l'Entorn 2. *Font: Elaboració pròpia*

La màquina M3 pateix una fallada/aturada a l'instant 2 u.t i amb una durada de 4 u.t. S'observen dos temps morts. El de M2 és degut a la restricció de precedència del J4: la O3 encara s'està processant a la M1 i, per tant, la O4 del J4 no pot iniciar-se just quan la M2 queda lliure a l'instant 10 u.t. El segon temps mort, a la M3, entre la O1 de J3 i la O1 de J5 és simplement degut a que l'instant d'arribada de J5 és a l'instant 10 u.t.

6.3.3 Entorn 3: N = 8 treballs i M = 4 màquines

Amb aquest últim entorn , on la dimensió del problema ja és considerable, s'ha volgut posar a proba l'algorisme. S'haurà de planificar un total de 18 operacions en quatre màquines. Les operacions, rangs d'arribades dels treballs i els temps de processament de les màquines són els

següents:

Treball	Operacions			Arribada
Id	1a	2a	3a	(u.t)
1	2	3	4	0
2	1	2	3	0
3	5	2	1	0
4	3	4	5	0
5	1	2	3	4-8
6	3	4	5	6-10
7	5	2	1	8-12
8	2	3	4	10-14

Taula 8: Operacions i arribades dels treballs de l'Entorn 3.

Màquina	Temps de processament (u.t)				
Id	Op.1	Op.2	Op.3	Op.4	Op.5
1	-1	3	2	2	2
2	4	2	-1	2	2
3	3	-1	4	3	4
4	2	2	2	-1	-1

Taula 9: Temps de processament de les màquines de l'Entorn 3

Ja que hi ha un augment de la dimensió de $|S| \cdot |A|$, el nombre d'episodis d'entrenament s'ha augmentat a 5 milions. El nombre d'estats visitats ha augmentat substancialment respecte l'Entorn 2, de sis-cents mil, a set milions i mig. Les mètriques obtingudes al llarg dels episodis són:

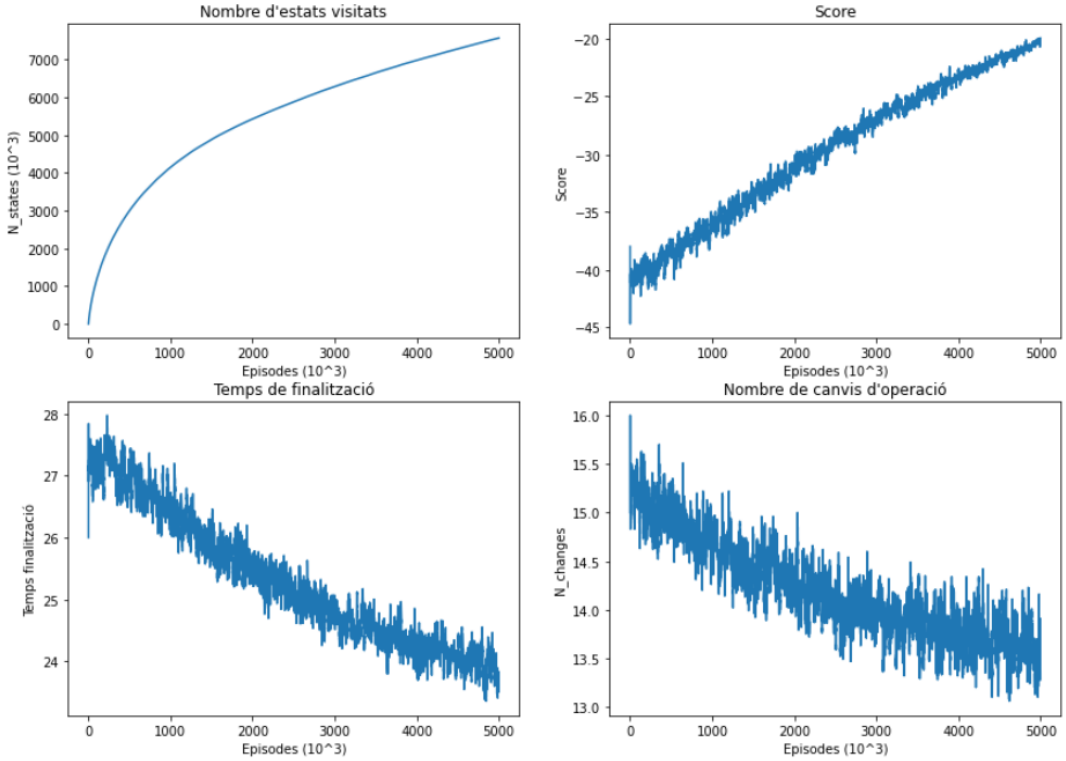

Figura 15: Mètriques de l'entrenament de l'Entorn 3 per a 5 milions d'episodis. *Font: Elaboració pròpia*

En aquest cas, l'entrenament realitzat de 5 milions d'episodis ha sigut clarament insuficient. El nombre d'estats visitats no s'ha estabilitzat, com sí ha passant a l'Entorn 2 (Figura 13). Això indica que, si s'hagués realitzat un entrenament més llarg, s'haurien visitat més estats. Tot i així, no s'ha realitzat un entrenament més extens per motius computacionals. El percentatge de memòria RAM ocupada després d'aquests cinc milions d'episodis era del 88% de la disponible. Qualsevol augment significatiu del nombre d'episodis (6 o 7 milions) s'hauria traduït en un error de memòria.

Tot i així, es presenta una simulació de planificació:

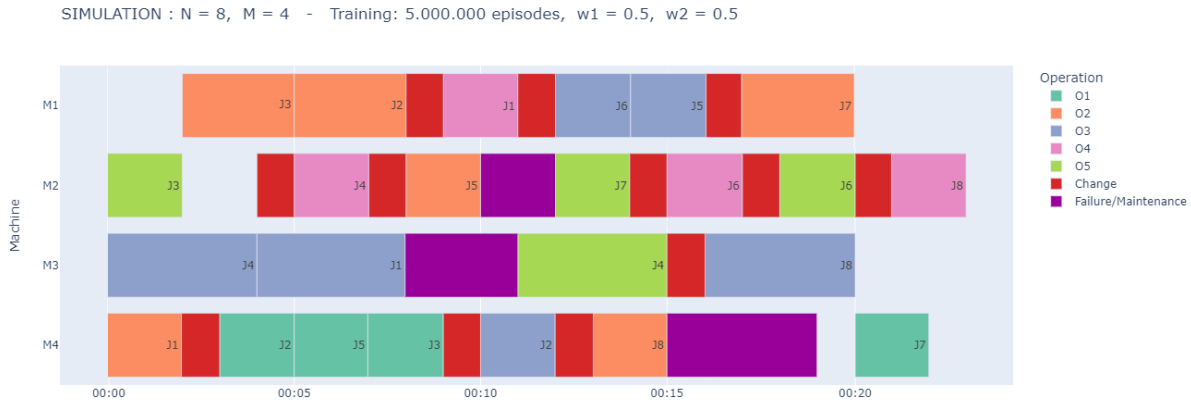

Figura 16: Simulació de planificació de l'Entorn 3. *Font: Elaboració pròpia*

S'observen força fallades de màquina i 3 temps morts tots ells induïts per restriccions de precedències. A simple vista, no s'observa una clara optimització dels canvis d'operació, com sí passava als entorns anteriors.

6.3.4 Discussió dels entrenaments realitzats

Un cop realitzats els entrenaments i obtinguts els resultats per als diferents entorns, es poden resumir els indicadors d'aquests entrenaments a la Taula 10. Donat que els entorns 2 i 3 s'han entrenat amb els paràmetres $\alpha = 0,2$, $w_1 = w_2 = 0.5$, es presenta el temps d'entrenament i estats visitats per a aquesta configuració de l'Entorn 1 (Cas 3). L'estabilitat i el rati d'estats visitats respecte els possibles estats és comú per a tots els casos de l'Entorn 1.

Entorn	N	M	Episodis	Temps	Estats visitats	Estats possibles	$\frac{\text{Estats visitats}}{\text{Estats possibles}}$	Estabilitat
1	4	2	2,5 M	1 : 06 : 58	20 849	$\approx 9,4 \cdot 10^6$	0,22 %	Sí
2	6	3	4 M	3 : 34 : 12	593 433	$\approx 2,9 \cdot 10^{10}$	0,002 %	Dubtosa
3	8	4	5 M	5 : 41 : 48	7 563 643	$\approx 3,4 \cdot 10^{13}$	0,00002%	No

Taula 10: Dades i indicadors dels entrenaments realitzats dels tres entorns estudiats ($\alpha = 0.2$, $w_1 = 0.5$, $w_2 = 0.5$). *Font: Elaboració pròpia*

La informació més rellevant que se'n pot extreure és, en primer lloc ,el creixement exponencial del nombre d'estats visitats i, en segon lloc, el decreixement, també exponencial, del rati d'estats visitats respecte els possibles estats representables. Com s'ha comentat, per a l'Entorn 1, s'assoleix una estabilitat del sistema: es visiten pràcticament tots els estats possibles i la solució no varia més enllà del primer milió d'episodis d'entrenament. Per a l'Entorn 2, l'assoliment de l'estabilitat del sistema ja és més dubtosa. És important notar que el rati d'estats visitats respecte el nombre d'estats representables disminueix de manera significativa de l'Entorn 1 a l'Entorn 2. És impossible determinar amb certesa quin hauria de ser aquest rati i si hauria de ser el mateix en qualsevol entorn. Ara bé, analitzant les mètriques d'entrenament de l'Entorn 2, és molt poc probable que, per a un entrenament més prolongat, s'assoleixi un nombre d'estats visitats tal que el rati esdevingui proper a 0,22 %. Implicaria visitar vora els 64 milions d'estats (11 vegades més). La corba de la Figura 13 presentaria un pendent molt més pronunciat i no s'estabilitzaria. Tot fa pensar doncs, que el nombre de possibles estats representables creix a un ritme molt més elevat que no el nombre d'estats realment accessible.

Pel que fa a l'Entorn 3, no se'n pot extreure gaire informació útil perquè l'entrenament ha sigut insuficient. Es podria haver assumit un temps d'entrenament major (més episodis) però la limitació ha vingut donada per la memòria que la taula de valors $Q(s, a)$ ocupava.

7 Planificació temporal

En un inici es va plantejar el desenvolupament d'aquest projecte en un marc de 4 mesos. Aquest s'inicià a finals de febrer, i es planificaren les tasques com es pot veure al diagrama de Gantt de la Figura 17, amb finals de juny com a data objectiu de finalització. Les tasques es divideixen en 3 grups o etapes diferenciades. La primera és la de definició del problema, documentació i tot el procés de recerca, tant de l'estat de l'art, com de l'aprenentatge per reforç. Seguidament s'inicià la segona etapa, on bàsicament es programà l'entorn i es desenvolupà la implementació de l'algorisme Q-*learning*. També inclou l'experimentació amb l'entorn i la presentació dels resultats obtinguts. La tercera fase correspondria a la redacció de la memòria. Aquesta es dona de manera intermitent durant la resta de tasques del projecte, sobretot durant la recerca i documentació i paral·lelament amb l'anàlisi dels resultats obtinguts.

Tanmateix, per motius personals i laborals, durant les dues últimes setmanes de maig i primera de juny, no es va poder dedicar el temps suficient al projecte i això ha fet que la planificació inicial es veiés dilatada. A més a més, algunes tasques s'han allargat més del previst. Amb tot això, la distribució temporal final és la que es pot veure a la Figura 18.

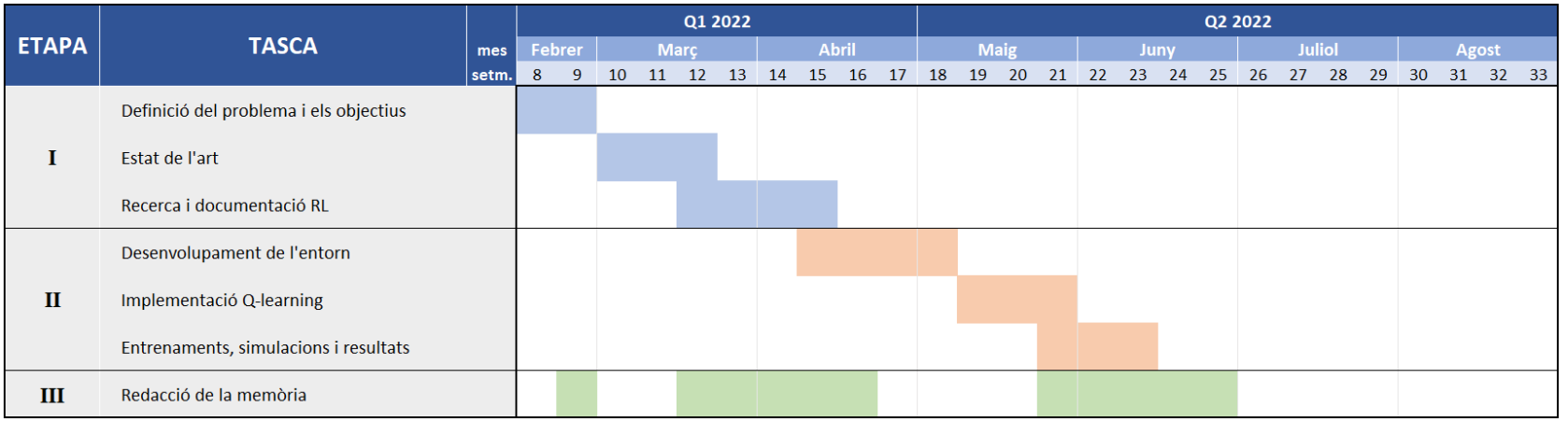

Figura 17: Diagrama de Gantt de la planificació inicial del projecte. *Font: Elaboració pròpia*

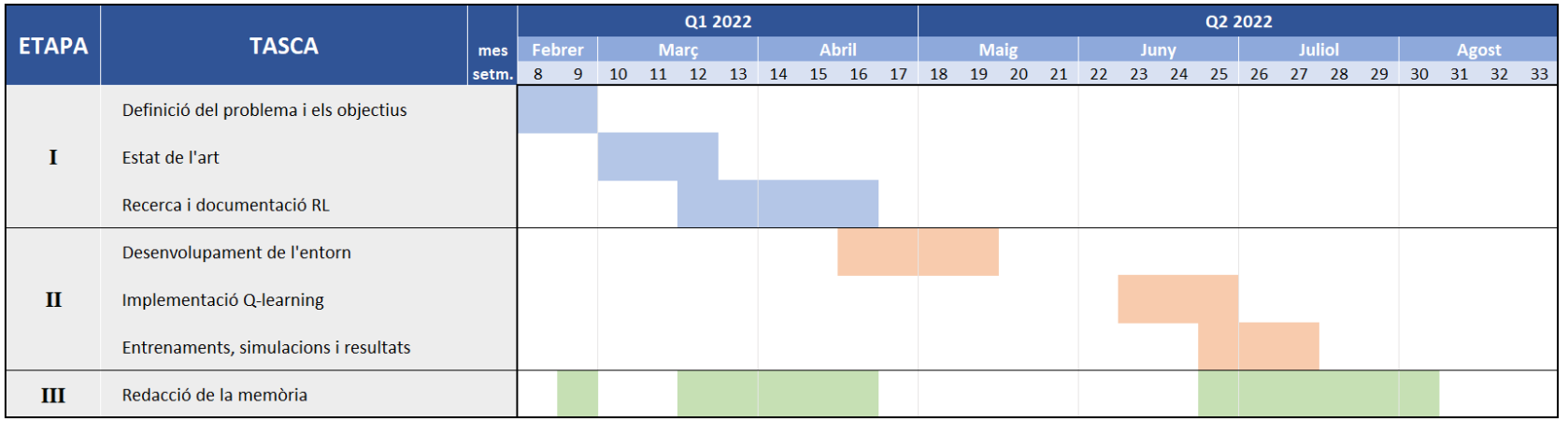

Figura 18: Diagrama de Gantt de la distribució temporal real del projecte. *Font: Elaboració pròpia*

8 Estudi econòmic

En aquesta secció es pretén estimar els costos econòmics que hauria comportat aquest projecte.

Aquest treball de final d'estudis de màster està reconegut amb 12 crèdits ECTS. Segons la normativa Europea, cada crèdit ECTS ha de tenir una càrrega de treball aproximada d'entre 25 i 30 hores. Això representa un total d'entre 300 i 360 hores. El valor de 360 hores és el que s'ajusta més al total d'hores invertides en aquest projecte. Aquest nombre d'hores serà important en el càlcul de diferents costos. Aquests costos es categoritzen segons la seva naturalesa.

8.1 Costos de personal

En primer lloc, es considera el cost de personal, és a dir, de l'autor del present document, que assumiria un rol de científic de dades o programador. Donat que el projecte correspon al treball final dels Estudis del Màster Universitari en Enginyeria Industrial (MUEI) es podria considerar com a salari la remuneració en concepte d'ajut a les pràctiques que el centre recomana per a aquests estudis: 10 € /h. A partir de les hores invertides comentades anteriorment:

$$\text{Cost de personal} = 360 \text{ hores} \cdot \frac{10\,€}{\text{hora}} = 3600\,€$$

Es podria analitzar el repartiment d'aquest cost entre les diferents tasques en funció de les hores dedicades a cada una d'elles.

Tasca	Dedicació (h)	Dedicació (%)	Cost (€)
Definició del problema i els objectius	4	1%	40
Estat de l'art	20	6%	200
Recerca i documentació RL	78	22%	780
Desenvolupament de l'entorn	88	24%	880
Implementació Q-learning	56	16%	560
Entrenaments, simulacions i resultats	30	8%	300
Redacció de la memòria	84	23%	840
Total	360 h	100%	3600 €

Aquelles tasques amb major dedicació i per conseqüent, cost, han estat el desenvolupament de l'entorn, la redacció de la memòria i la recerca i documentació sobre aprenentatge per reforç, seguit de la implementació del codi de l'algorisme Q-*learning*.

8.2 Costos de subministraments

En aquest cas, es considera el consum elèctric generat durant aquestes 360 hores. Els elements que han generat aquest consum han estat l'ordinador portàtil i el monitor usat com a segona

pantalla. El portàtil consumeix 44 kW ([33]) i el monitor 15 kW. També es pot tenir en compte la il·luminació de l'espai de treball (20 W) amb un factor corrector de 0,3, ja que aproximadament el 70% de les hores treballades eren diürnes i no era necessària il·luminació artificial. El preu del kWh que es té contractat és de 0,22 € /kWh. Així doncs el cost del subministrament elèctric imputable al projecte és:

$$\text{Cost subministrament} = (44\ \text{W} + 15\ \text{W} + 0,3 \cdot 20\text{W}) \cdot 360\ \text{h} \cdot 0,22\ \frac{€}{\text{kWh}} = 5,15\,€$$

En aquesta partida també s'hi pot incloure el cost corresponent a la connexió a Internet. La tarifa que es paga és de 29,99 € mensuals. Això significa un cost de 29,99 € /(30 dies · 24 h) = 0,042 €/h. El cost imputable al projecte és:

$$\text{Cost connexió Internet} = 0,042\frac{€}{\text{h}} \cdot 360\text{hores} = 15\,€$$

8.3 Costos d'amortitzacions

Per últim, es considera el cost d'amortització dels equips utilitzats. Com s'ha dit, s'usa un portàtil de valor de compra de 900 € i un monitor de 120 €. A més a més, s'usa un teclat i un ratolí de 30 €. Es considera una vida mitjana de tots aquests dispositius electrònics de 6 anys. Si la durada del projecte ha estat d'uns 5 mesos:

$$\text{Costos d'amortització} = (900\,€ + 120\,€ + 30\,€) \cdot \frac{5\ \text{mesos}}{6\ \text{anys} \cdot 12\ \text{mesos}} = 72,9\,€$$

8.4 Cost total

Més enllà dels descrits, no es consideren altres costos. Per exemple, referent al *software* emprat, tant Overleaf (editor de text en llenguatge LaTeX) com Jupyter Notebook i PyCharm (editors de codi Python) i GitLab (repositori online) són de caràcter gratuït.

Cost	Valor (€)	Percentatge(%)
Personal	3600,00	97,5%
Subministrament elèctric	5,15	0,1%
Servei de connexió a Internet	15,00	0,4%
Amortitzacions	72,90	2,0%
Total	3693,05 €	100%

Taula 11: Resum de costos. *Font: Elaboració pròpia*

Resumint, els costos totals ascendeixen a 3 693,05 €. Pràcticament tota la despesa és en concepte de despesa de personal.

9 Estudi d'impacte ambiental

Per la naturalesa d'aquest projecte, de recerca i programació, l'impacte ambiental que aquest presenta és limitat. En primer lloc, caldria considerar les emissions de CO_2 fruit de la producció de l'energia elèctrica consumida pels equips utilitzats. Ara bé, la companyia contractada pel subministrament elèctric assegura que el 100% de l'electricitat prové de fonts renovables. Així doncs, l'impacte d'aquest consum és molt reduït. D'altra banda, sí que s'haurien de tenir en compte les emissions i els materials usats en la fabricació dels dispositius electrònics, especialment el portàtil i el monitor. Serà important que un cop aquests arribin al final de la seva vida útil, es reciclin de manera adequada. De totes maneres cap d'aquests dispositius s'ha adquirit de manera expressa per a la realització d'aquest projecte i la seva vida útil va molt més enllà d'aquest. Finalment, comentar que un dels objectius perseguits amb aquest estudi era minimitzar els canvis d'operació en una màquina. Més enllà del cost temporal que poden suposar, depenent del sistema productiu, pot implicar un desaprofitament o merma de material que, a més a més d'un cost econòmic, representa un cert impacte ambiental. Si la solució posposada es pogués aplicar en un entorn real, de ben segur tindria un impacte positiu pel que fa a la dimensió ambiental.

Conclusions

En un inici es va plantejar aquest projecte amb la voluntat d'abordar el problema d'assignació d'operacions des d'una vessant no treballada fins al moment. Per al problema estàtic, sense esdeveniments estocàstics, multitud de mètodes i heurístiques s'han implementat donant bons resultats. Quan es vol aproximar el problema a un entorn real, on la dinàmica del sistema no està totalment sota el control de qui pren les decisions, el nombre d'algorismes i mètodes existents es redueix substancialment. Repassant la literatura disponible, diversos mètodes aproximats s'han proposat per resoldre aquest DJSSP més desafiant. Des del RL, la majoria d'implementacions són amb Q-*learning* i Deep Q-*learning* (DQN). Tot i així s'ha pogut comprovar, de primera mà, la limitació que un mètode tabular com el Q-*learning* presenta enfront a un problema que augmenta la seva complexitat de forma exponencial davant d'un increment lineal i moderat del nombre de màquines o treballs. S'ha vist el potencial del Q-*learning* per trobar bones solucions en entorns dinàmics senzills, quan el nombre de treballs i màquines es mantenen baixos, amb temps d'entrenament raonables. Quan el nombre d'estats i/o accions possibles creix de manera desmesurada, aquest mètode deixa de ser efectiu i es posa de manifest la necessitat d'un mètode aproximat que, tot i no garantir la solució òptima, permeti trobar solucions aproximades amb un temps i cost computacional acotat.

Amb aquest projecte s'aconsegueix una bona introducció al món de l'aprenentatge per reforç, molt menys popular que l'aprenentatge supervisat i no supervisat. Es pren consciència de les possibilitats que aquest ofereix. Tot i que aquest treball acadèmic hagi arribat al seu final, es seguirà investigant, com a projecte personal, altres mètodes per abordar el DJSSP en entorns més complexos. Mètodes aproximats que permetin treballar amb grans espais d'estats i puguin oferir una solució a aquells entorns on els mètodes tabulars, com el Q-*learning*, no hi arriben. Tot i que la implementació proposada no hagi pogut entrenar i proposar solucions a entorns més grans i complexos dels presentats, el balanç que es fa del projecte és positiu i satisfactori.

Bibliografia

[1] Mayra Georgina Miranda Flores, *Implementación de un algoritmo de recocido simulado para el problema de planificación de Tareas en una Empresa de Manufactura de Cosméticos*,LIBRARY, (nov. de 2009) pàg. 39-43
Consultat (25-3-2022)

[2] OpenGenius IQ, *Job Shop Problem*, URL: https://iq.opengenus.org/job-shop-problem/
Consultat (4-3-2022)

[3] Yang, Yoo Baik, *Methods and Techniques Used for Job Shop Scheduling*, RETROSPECTIVE THESES AND DISSERTATIONS (1977)
Consultat (4-3-2022)

[4] Jatoth Mohana, Krishnanand Lankab,A. Neelakanteswara Rao, *A Review of Dynamic Job Shop Scheduling Techniques*, ELSEIVER, (2019)
Consultat (5-3-2022)

[5] Fox M.S., Smith S.F., *ISIS: A knowledge based system for factory scheduling*, EXPERT SYSTEMS, (1984)
Consultat (16-3-2022)

[6] Christian Bierwirth, Dirk C.Mattfeld, *.Production Scheduling and Rescheduling with Genetic Algorithms*, . EVOLUTIONARY COMPUTATION, (1999)
Consultat (16-3-2022)

[7] Wikipedia, *Simulated annealing*, https://en.wikipedia.org/wiki/Simulated_annealing
Consultat (17-3-2022)

[8] Wikipedia, *Ant colony optimization algorithms*, URL: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms
Consultat (17-3-2022)

[9] Hu Yan-hai, Yan Jun-qi, Ye Fei-fan i Yu Jun-he , *Flow shop rescheduling problem under rush orders*, JOURNAL OF ZHEJIANG UNIVERSITY, SCIENCE, (oct. de 2005)
Consultat (17-3-2022)

[10] Machine Learning Mastery, *A Gentle introduction to particle swarm optimization*, URL: https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/
Consultat (19-3-2022)

[11] Wang Ce, Wang Shu-feng, Feng Dong-qing, *Application of hybrid PSO in job-shop dynamic scheduling problem*, COMPUTER ENGINEERING AND APPLICATIONS, (2010)
Consultat (19-3-2022)

[12] Yu Jian-jun, Sun Shu-dong, Wang Jun-Qiang, *Flexible Manufacturing Cell Dynamic Scheduling by Immune Algorithm* , ACTA AAERONAUTICA ET ASTRONAUTICA SINICA, (2007)
Consultat (20-3-2022)

[13] Min-Jung Yoo, Jean-Pierre Müller, *Using Multi-Agent System for Dynamic Job Shop Scheduling*, RESEARCH GATE, (gen. 2002)
Consultat (2-4-2022)

[14] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, *Playing Atari with Deep Reinforcement Learning*, ARXIV, URL: https://arxiv.org/abs/1312.5602
Consultat (2-4-2022)

[15] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis, *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*, ARXIV, URL: https://arxiv.org/abs/1712.01815
Consultat (2-4-2022)

[16] Vinyals, O., Babuschkin, I., Czarnecki, W.M. et al., *Grandmaster level in StarCraft II using multi-agent reinforcement learning*, NATURE 575, (2019) pàg. 350–354
Consultat (2-4-2022)

[17] Chang, J.; Yu, D.; Hu, Y.; He, W.; Yu, H, *Deep Reinforcement Learning for Dynamic Flexible Job Shop Scheduling with Random Job Arrival*, PROCESS, (2022)
Consultat (23-3-2022)

[18] Wang, S.; Sun, S.; Zhou, B.; Xi, L.F, *Q-Learning Based Dynamic Singe Machine Scheduling*, J. SHANG HAI JIAO TONG UNIV, (2007)
Consultat (26-3-2022)

[19] Fonseca-Reyna, Y.C.; Martinez, Y.; Rodríguez-Sánchez, E.; Méndez-Hernández, B.; Coto-Palacio, L.J., *An Improvement of Reinforcement Learning Approach to Permutational Flow Shop Scheduling Problem*, ICOR, (2018)
Consultat (26-3-2022)

[20] Shahrabi, J.; Adibi, M.A.; Mahootchi, M., *A reinforcement learning approach to parameter estimation in dynamic job shopscheduling*, COMPUT. IND. ENG., (2017)
Consultat (26-3-2022)

[21] Wang, Y.-F., *Adaptive job shop scheduling strategy based on weighted Q-learning algorithm*, . J. INTELL. MANUF, (2018)
Consultat (26-3-2022)

[22] Wang, H.; Sarker, B.R.; Li, J.; Li, J., *Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning*, INT. J. PROD. RES., (2020)
Consultat (27-3-2022)

[23] Bouazza, W.; Sallez, Y.; Beldjilali, B, *A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect.*, IFAC PAP, (2017)
Consultat (27-3-2022)

[24] Luo, S., *Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning*, APPL. SOFT COMPUT, (2020)
Consultat (27-3-2022)

[25] Luo, S.; Zhang, L.; Fan, Y. , *Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning*, COMPUT. IND.ENG, (2021)
Consultat (27-3-2022)

[26] Longfei Zhou, Lin Zhang, Berthold K.P. Horn, *Deep reinforcement learning-based dynamic scheduling in smart manufacturing*, ELSEIVER, (2020)
Consultat (16-4-2022)

[27] M. Emin Aydin, Ercan Öztemel, *Dynamic job-shop scheduling using reinforcement learning agents*, ELSEIVER, (1999)
Consultat (16-4-2022)

[28] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, 2a edició, (2018), URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
Consultat (2-8-2022)

[29] Wikipedia, *Andrey Markov*, URL: https://en.wikipedia.org/wiki/Andrey_Markov
Consultat (12-7-2022)

[30] Watkins, C.J.C.H., Dayan, P., *Q-learning*,MACH LEARN 8, pàg. 279-292 (1992) 2a edició, (2018), URL: https://doi.org/10.1007/BF00992698
Consultat (12-7-2022)

[31] OpenAI, *Gym Documentation*, URL: https://www.gymlibrary.dev/
Consultat (1-8-2022)

[32] Plotly.Express, *Timeline Documentation*, URL: https://plotly.com/python-api-reference/generated/plotly.express.timeline.html
Consultat (30-7-2022)

[33] NoteBookCheck, *Review del Convertible ASUS ZenBook Flip 14 UM462DA*, URL: https://tinyurl.com/y64fn7ah
Consultat (30-7-2022)

[34] GitLab, *DJSSP with RL*, URL: https://gitlab.com/mhernandezcamp/djssp-with-rl