REQUERIMENTS EN INTEL·LIGÈNCIA ARTIFICIAL PER L'ALLOTJAMENT EFICIENT DE XARXES VIRTUALS

Treball final de grau
Desenvolupat a
L'escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
Per
David Escobar Pérez

En compliment parcial
dels requisits per obtenir el grau en
ENGINYERIA DE TECNOLOGIES I SERVEIS DE LA TELECOMUNICACIÓ

Tutor: Xavier Hesselbach Serra

Barcelona, Setembre de 2021

Resum

La virtualització de xarxes està guanyant cada cop més importància, ja que permet adequar les necessitats d'una xarxa a noves circumstancies, el que deriva en una flexibilitat major. La virtualització de la xarxa depèn en gran mesura de l'optimització en l'assignació dels recursos de la xarxa física, esdevenint en un dels grans problemes de l'actualitat. Aquest problema rep el nom d'allotjament de xarxes virtuals (VNE), molts algorismes s'han utilitzat per resoldre aquest problema d'allotjament de recursos obtenint resultats pobres en termes de revenue. Degut a l'auge que hi ha en l'actualitat amb la intel·ligència artificial com a medi per a resoldre els problemes tecnològics dels últims anys, és proposa la utilització de l'algorisme Q learning, un algorisme típic de l'aprenentatge de reforç. Els resultats obtinguts demostren una millora en la utilització de recursos, el que permet obtenir millors valors en termes de Cost/Revenue, comparat amb altres algorismes tradicionals.

Resumen

La virtualización de redes está ganando cada vez más importancia, ya que permite adecuar las necesidades de una red a nuevas circunstancias, lo que deriva en una flexibilidad mayor. La virtualización de la red depende en gran parte de la optimización en la asignación de recursos de la red física, lo que lo convierte en uno de los grandes problemas de la actualidad. Este problema recibe el nombre de alojamiento de redes virtuales (VNE), muchos algoritmos se han utilizado para resolver este problema de alojamiento de recursos obteniendo resultados pobres en términos de Revenue. Debido al auge que hay en la actualidad con la inteligencia artificial como medio para resolver los problemas tecnológicos de los últimos años, se propone la utilización del algoritmo Q learning, un algoritmo típico del aprendizaje por refuerzo. Los resultados obtenidos demuestran una mejora en la utilización de recursos, lo que permite obtener mejores valores en términos de Cost/Revenue comparado con otros algoritmos tradicionales.

Abstract

Network virtualization is becoming increasingly important, as it allows to adapt the needs of a network to new circumstances, resulting in greater flexibility. Network virtualization depends mostly on the optimization of the allocation of physical network resources, becoming one of the major problems of today. This problem is called virtual network embedding (VNE), many algorithms have been used to solve this virtual network embedding problem with poor results in terms of revenue. Due to the current rise of artificial intelligence, it has been used in order to solve technological problems of recent years, the use of the Q learning algorithm, a typical algorithm for reinforcing learning, is proposed. The results achieved show an improvement in the use of resources which allows to obtain better values in terms of Cost/Revenue, compared to other traditional algorithms.

Contingut

Llista de figures

Llista de taules

1. Introducció

1.1. Motivació

Un dels majors reptes a la virtualització de xarxes és trobar la forma més efectiva d'allotjar les demandes de les diferents xarxes virtuals sobre una xarxa física intentant tenir el millor aprofitament dels recursos. Aquest problema és l'anomenat Virtual Network Embedding (VNE).

Avui dia, l'ús de la Intel·ligència Artificial (IA) com a medi per a resoldre problemes complexes de caire tecnològic ha anat augmentant any rere any. Això obre un ventall de possibilitats per poder afrontar problemes com el de VNE, que no s'havien plantejat fins ara tenint en compte els algorismes d'IA. Aquest escenari es considera com un gran repte de cara a poder resoldre el problema de VNE a partir dels recursos que és poden obtenir de la IA.

1.2. Objectiu

El principal objectiu d'aquest projecte és obtenir una solució al problema del VNE fent ús d'eines provinents de la IA. Per aquest propòsit, s'haurà de dur a terme una recerca exhaustiva dels diferents tipus d'aprenentatges, trobar el més òptim, escollir l'algorisme més adient del tipus d'aprenentatge escollit i adaptar-ho al problema que esdevé el VNE.

1.3. Definició del problema

Aquest projecte pretén trobar respostes per als següents punts:

- Quin és el problema que s'enfronta al VNE?
- Pot ajudar la IA a resoldre el problema de VNE?
- Quins aprenentatges són els més òptims per a la seva utilització com a resolució del problema?
- Quins són els resultats obtinguts de l'algorisme de IA emprat al VNE? Tests i comparacions.

1.4. Enfocament de la memòria

L'enfocament que se li ha donat a aquesta memòria és el següent:

1. Comprovar i estudiar el background i les tecnologies utilitzades al VNE.
2. Comprovar i estudiar el background del tipus d'aprenentatges de la IA.
3. Escollir un algorisme de IA i elaborar un pseudocodi que resolgui el problema del VNE.
4. Cercar un software (entorn) per a la implementació de l'algorisme i fer-ne proves i tests.
5. Comparar els resultats obtinguts amb altres algorismes presents al programa.
6. Conclusions i marge de millora.

1.5. Esbós de la memòria

L'estructura que se li ha donat a la memòria és la següent:

- Capítol 1 Introducció del treball.
- Capítol 2 Descripció dels principals conceptes empleats a la memòria.
- Capítol 3: Recerca de l'entorn i de l'algorisme més adient per a resoldre el problema.
- Capítol 4 Descripció de la implementació de l'algorisme .
- Capítol 5 Exposició dels resultats.
- Capítol 6 Presentació del pressupost del treball.
- Capítol 7 Exposició de les conclusions del treball i el treball futur.

2. Background

En aquest capítol el que es pretén és fer un breu resum sobre els principals conceptes teòrics que es fan servir al llarg del treball a fi de poder establir les bases adequades per a la seva comprensió.

2.1. Xarxes virtuals

Una xarxa virtual [1] és una xarxa en la que tots els aparells, servidors, màquines virtuals i centres de dades connectades entre si són constituïdes mitjançant software.

Tots els instruments que interactuen entre ells a la xarxa ho fan mitjançant la tecnologia d'internet, permetent obtenir xarxes molt més amplies que si fossin cablejades.

2.1.1. Xarxes superposades

Una xarxa superposada[2] és construïda sobre una xarxa existent, generalment utilitzant tecnologies d'encapsulament i de túnels. El major avantatge d'aquest tipus de xarxa és el de poder implementar una nova xarxa molt econòmica fent servir la infraestructura d'una ja existent.

2.1.2. Xarxes privades virtuals (VPNs)

És un conjunt de xarxes privades [2] que es connecten entre elles però són separades de xarxes públiques com internet. Generalment les organitzacions empren aquests tipus de xarxes per connectar les seves oficines ubicades geogràficament a diferents llocs.

2.1.3. Xarxes virtuals compartides (VSN)

Aquesta tecnologia[2] permet compartir els recursos físics entre múltiples instàncies de xarxes. Les diferents instàncies de la xarxa poden compartir el mateix punt d'accés switches físics, routers i servidors.

Tecnologia	Èmfasi	Xarxa virtual
Superposada	Servei nou	Nodes conscients del servei
VPN	Connectivitat	Annexar xarxes privades
VSN	Compartir recursos	Part de la infraestructura física

Taula 1: Comparació de les diferents tecnologies de xarxes virtuals [2]

2.2. Intel·ligència artificial

Una de les característiques de la intel·ligència artificial és la seva capacitat d'aprendre automàticament. Aquesta depèn del tipus d'aprenentatge automàtic (Machine learning) que estem utilitzant.

2.2.1. Machine learning

Machine learning[3] (ML) o aprenentatge autònom és un camp de la informàtica que s'ocupa de la creació d'algorismes, que es basen en una col·lecció d'exemples sobre algun tipus de fenomen. Es poden distingir quatre tipus de Machine learning, segons la supervisió que necessiten: supervisat, no supervisat, semi-supervisat i per reforç.

2.2.1.1. Aprenentatge supervisat

En l'aprenentatge supervisat [3] el conjunt de dades és la col·lecció d'exemples etiquetats $\{(x_i,y_i)\}_{i=1}^{N}$. Cada element x_i és un vector de característiques pertanyents al conjunt de dades, pot ser un conjunt finit de classes o simplement un número real. Cada y_i pot ser un element que pertany a un conjunt finit de classes, un numero real o una estructura més complexa, on una classe és una categoria a la qual pertany un exemple.

L'objectiu dels algorismes d'aprenentatge supervisat és fer ús del conjunt de dades per produir un model que prengui el vector de característiques x com a informació d'entrada i sortida per tal de que es pugui deduir l'etiqueta per aquest vector de característiques. Als següents apartats es podrà veure de forma molt resumida els principals tipus d'algorismes que existeixen dintre de l'aprenentatge supervisat.

2.2.1.1.1. Algorismes de regressió

Quan s'utilitzen algorismes de regressió com: Regressió lineal o no lineal, màquines de vectors de suport (SVM), arbres de decisió, boscs aleatoris o xarxes neuronals [3]. El resultat que s'obté al aplicar aquests algorismes és un número dintre d'un conjunt infinit de

possibilitats. Tenint en compte aquesta dada, aquest tipus d'algorismes es poden aplicar en problemes semblants als següents:

- Predir els valors en els que es pot vendre un immoble.
- Predir els productes que pot vendre una empresa.
- Predir el temps necessari en completar un trajecte.

2.2.1.1.2. Algorismes de classificació

Quan s'utilitzen algorismes de classificació com: Regressió logística, màquines de vector de suport (SVM), arbres de decisió o xarxes neuronals [3]. El resultat que s'obté és un element de tipus categòric entre un número limitat de categories. És a dir, en el cas de que es volgués detectar si un correu fos spam o no spam i s'escollís un algorisme de classificació aquest podria obtenir dos possibles resultats, hauria d'escollir si el correu pertany a la categoria spam o pertany a la categoria no-spam.

Dintre d'aquests tipus d'algorismes, molts poden donar el resultat de la categoria a la que es pertany amb una probabilitat, en aquests casos sol escollir-se la categoria donada amb una probabilitat més gran. Alguns dels exemples en els que es pot aplicar aquests tipus d'algorismes son les següents:

- En el context d'un aparell electrònic: Quin esport està fent el portador? [bici, córrer, nedar, corda].
- En un ordinador en la detecció de virus: [si, no]
- En el context d'activitats de màrqueting: Serà aquest client fidel a la companyia? [si, no]

2.2.1.2. Aprenentatge no supervisat

A l'aprenentatge no supervisat [3] el conjunt de dades és una col·lecció d'exemples sense etiquetar $\{(x)\}_{i=1}^{N}$. L'objectiu d'aquest d'aprenentatge és crear un model que agafi el vector de característiques **x** com a entrada i el transformi en un altre vector o valor útil per resoldre un problema pràctic. Es vol trobar grups similars entre els conjunts de dades.

L'agrupament es podrà considerar com un dels principals problemes existents en aquest tipus d'aprenentatge; es tracta de trobar una estructura en la col·lecció de dades sense etiqueta. Les estructures de dades, que són similars entre si i que són diferents a altres dades pertinents a altres estructures s'anomenen clústers.

Clustering és el procés d'organitzar les dades en grups o estructures en les que són similars d'alguna manera. Algunes de les aplicacions de Clustering més conegudes poden ser:

- Trobar grups homogenis: Reprenent l'exemple de l'ús per a qüestions relatives al màrqueting, es podria trobar un grup de clients amb les mateixes necessitats per vendre el producte o servei que l'empresa proveeix.

- Trobar grups i descriure'ls en base a les seves propietats: Com podria ser a la classificació de malalties rares en l'àmbit de la medicina.

- Detecció de casos anòmals: Per exemple amb un sistema de detecció de fraus de targetes de crèdit.

Als següents apartats es podrà veure d'una forma esquemàtica els principals tipus d'algorismes d'agrupació que existeixen dintre de l'aprenentatge no supervisat.

2.2.1.2.1. Agrupament exclusiu

En aquests algorismes, els agrupaments[4] es poden generar per mitjans que desglossen les dades creant un número de k determinats clústers. Generalment s'utilitzen mesures de distància per a generar els clústers.

- Cada un dels clústers té com a mínim un objecte.

- Cada objecte pot pertànyer a un sol clúster. L'exemple més representatiu és l'algorisme K-means.

2.2.1.2.2. Superposició de clústers

En aquests tipus d'algorismes s'empren conjunts difusos per agrupar dades de forma que cada punt pot pertànyer a dos o més clústers en diferents graus d'incidència. En aquest cas, les dades s'associaran amb un valor apropiat depenent del grau d'incidència. L'algorisme Fuzzy C-means [4] genera aquests clústers difusos.

2.2.1.2.3. Agrupament jeràrquic

Algorismes que creen una estructura jeràrquica[4] de clústers mitjançant la creació de clústers al llarg d'iteracions. Existeixen dues maneres diferents d'arribar a aquesta jerarquia.

De forma divisòria, en la qual es parteix amb un de sol clúster en el primer nivell, i a partir d'ell es van construint clústers en els següents nivells.

De forma acumulativa, en la qual es parteix de nombrosos clústers més petits i aquests es van agrupant progressivament fins a generar estructures jeràrquiques que arriben fins al primer nivell.

2.2.1.2.4. Agrupacions probabilístiques

Fa servir el caire probabilístic. Un algorisme que s'utilitza en aquest tipus és **l'**algorisme d'expectació-Maximització (EM) [4]**.**

2.2.1.3. Aprenentatge de reforç

Aprenentatge per reforç [5] és una branca de ML que s'encarrega d'entrenar els models per a poder prendre una seqüència de decisions. Aquesta branca es pot entendre com un model ben definit (figura 1). El model està format per un agent intel·ligent i un entorn, l'agent rep l'estat actual de l'entorn, llavors pren una acció per poder passar al següent estat, un cop presa la acció es notifica al entorn i aquest li torna una recompensa segons si l'acció escollida ha estat encertada o no i canvia d'estat.

L'objectiu final del Reinforcement Learning (RL) és aprendre una estratègia d'accions que permeti escollir la que el sistema determini que obtindrà una recompensa acumulada més gran. Quan un entorn es modelat com Markov, la interacció entre l'agent i l'entorn s'ha de veure com un procés de decisió de Markov.

Aquest procés es pot definir amb quatre factors (S,A,R,P); entenent S com el conjunt d'estats de l'entorn, A com el conjunt d'accions, R: S X A-> R com a la funció de recompenses i P: S X A->P com a la probabilitat de canvi d'estat.

Degut a que les funcions P i R es basen en un conjunt d'accions i estats, són desconegudes per l'agent. Per tant, el sistema només pot decidir-se per estratègies referents a recompenses instantànies obtingudes arrel de les proves fetes cada vegada; s'ha de considerar la incertesa de l'entorn i els objectius a llarg termini seleccionant estratègies a l'hora d'escollir una acció, el que fa que s'hagi de definir una funció de valor entre la estratègia i la recompensa immediata:

$$V^{\pi}(s) \leftarrow \sum_{a \in A(s)} \pi(s,a) \sum_{s'} P^{a}_{ss'}[R^{a}_{ss'} + \gamma V^{\pi}(s')]$$

On $P^{a}_{ss'}$ és la probabilitat obtinguda pel sistema quan l'estat de l'entorn canvia de s a s' degut a l'acció a, $R^{a}_{ss'}$ és la recompensa instantània extreta pel sistema per canviar de l'estat s a s' degut a l'acció a i ɣ és un factor de descompte.

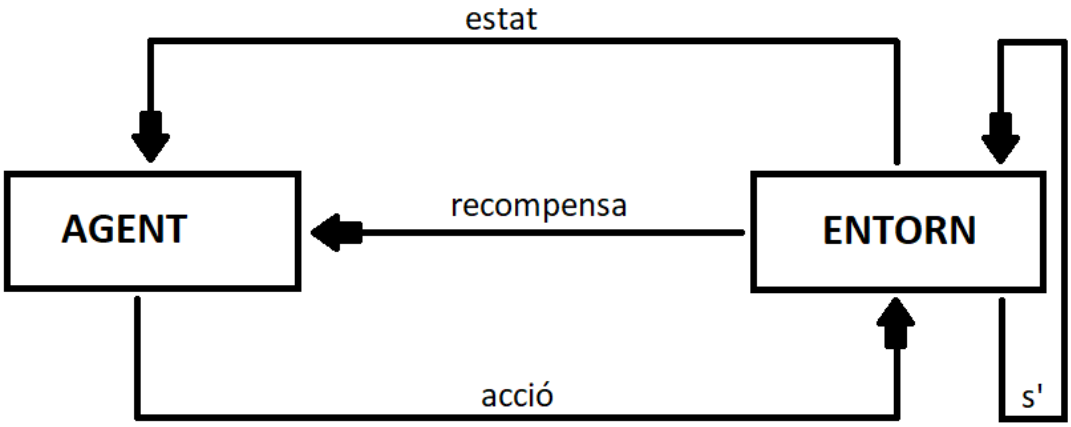

Figura 1: Model bàsic Reinforcement Learning [5]

Els algorismes típics de reforç basats en el model de pressa de decisions de Markov es poden dividir en dos tipus. D'una banda els que es basen en el model, com l'algorisme de SARSA. En aquests el que primer s'aprèn és el coneixement sobre el model i després se'n deriva l'estratègia òptima d'aquest coneixement. D'altra banda també hi ha aquells en els que el model és irrellevant, com l'algorisme de TD i Q-learning. En aquests últims, directament es calcula l'estratègia òptima sense coneixement del model.

2.2.2. Xarxes neuronals i aprenentatge profund (deep learning)

Una xarxa neuronal [3] és una funció matemàtica: $y=f_{nn}(x)$. Aquesta és una funció niada, en la que cada una és una capa diferent en el model. En conclusió, l'estructura d'una de les funcions niades és: $fl(z) = gl(Wlz + bl)$. Amb gl com a qualsevol funció matemàtica que sigui diferenciable, sovint es pot escollir la tangent hiperbòlica , Wl una matriu, z el vector d'entrada i bl un vector.

El concepte d'aprenentatge profund fa referència a entrenar les xarxes neuronals amb més de dos capes intermèdies.

2.2.2.1. Xarxa neuronal convolucional

Aquesta xarxa neuronal [3] redueix el nombre de paràmetres en xarxes neuronals amb més de dues capes sense haver de perdre massa qualitat del model emprant un filtre de convolució. S'utilitza sovint en imatges.

2.2.2.2. Xarxa neuronal recurrent

Aquesta xarxa neuronal [3] es fa servir per etiquetar, classificar o generar seqüències; és a dir, predir la classe per una seqüència sencera. Aquestes xarxes s'utilitzen en el processament de text degut a que són seqüències de paraules o caràcters.

2.2.3. Propietats dels principals algorismes

- **Algorisme K-means:** Pertany al tipus d'aprenentatge no supervisat. El seu objectiu és trobar i agrupar en classes els punts de dades que tenen una alta similitud entre ells.
    - PRO: Molt eficient des del punt de vista de CPU.
    - CONTRA: No són molt bons per a identificar classes quan es tracta de grups que no tenen una forma de distribució esfèrica.
- **Agrupació jeràrquica:** Pertany al tipus d'aprenentatge no supervisat. No necessita especificar el nombre d'agrupacions, les trobarà per si mateix.
    - PRO: Són especialment potents quan el conjunt de dades conté relacions jeràrquiques reals.
    - CONTRA: Són molt exigents des del punt de vista CPU.
- **Models de barreja Gaussiana:** Pertany al tipus d'aprenentatge no supervisat. Són models probabilístics que assumeixen que totes les mostres són generades a partir d'una barreja d'un nombre finit de distribució gaussiana amb paràmetres desconeguts.
    - PRO: Algorisme més ràpid en aprendre models de barreges, també té una gran flexibilitat en el número i la forma dels grups.
    - CONTRA: És molt sensible als valors inicials, que condicionen en gran mesura el seu rendiment.
- **Algorisme de regressió lineal:** Pertany al tipus d'aprenentatge supervisat. Pot servir quan es tenen dades numèriques amb moltes característiques o per realitzar prediccions econòmiques o de màrqueting.
    - PRO: Fàcil d'entendre i d'explicar, fàcil de modelar i menys propens al sobre ajust.
    - CONTRA: No es poden modelar relacions complexes, tampoc es poden capturar relacions no lineals sense transformar l'entrada.

- **Màquines vectorials de suport (SVM):** Pertany al tipus d'aprenentatge supervisat. Es fan servir a la classificació, regressió i detecció de valors atípics.
    - PRO: Eficaç en espais de grans dimensions. Utilitza un subconjunt de punt s d'entrenament en la funció de decisió, eficient en memòria.
    - CONTRA: No té una solució general per als problemes no lineals.
- **K-Nearest Neighbour:** Pertany al tipus d'aprenentatge supervisat. Aquest manté tots els exemples d'entrenament a la memòria. Un cop arriba un exemple no vist per l'algorisme, aquest troba els k exemples d'entrenament més propers a x i retorna l'etiqueta majoritària.
    - PRO: Simple i efectiu. Adequat per la classificació de classes amb mides de mostres grans.
    - CONTRA: Aprenentatge lent. La quantitat de càlculs és molt gran.
- **Arbres de decisió:** Pertany al tipus d'aprenentatge supervisat. S'utilitzen a la resolució de problemes de regressió i classificació.
    - PRO: Pot obtenir bons resultats i efectius per a grans fons de dades en un temps relativament curt.
    - CONTRA: Aparició de problemes de sobre-ajust i dificultat per a tractar amb dades que falten.
- **Xarxes neuronals artificials:** Pertanyen al tipus d'aprenentatge supervisat. Són models simples que emulen el funcionament del sistema nerviós d'un organisme viu.
    - PRO: Precisió de classificació alta, forta capacitat de processament en paral·lel, forta capacitat d'emmagatzematge i aprenentatge distribuït. Pot aproximar relacions complexes no lineals.
    - CONTRA: Requereixen un gran nombre de paràmetres, el temps d'aprenentatge és molt llarg.
- **Algorismes Q learning:** Pertanyen al tipus d'aprenentatge de reforç, concretament al que són lliures de model. Directament es calcula l'estratègia òptima sense coneixement previ del model emprat.
    - PRO: Aprèn una política òptima.
    - CONTRA: Agent agressiu, agafa el camí òptim amb risc a caure.
- **Algorismes SARSA:** Pertanyen al tipus d'aprenentatge de reforç, concretament als que són basats en un model. Primer s'aprèn el coneixement del model i després en

deriva l'estratègia optima del coneixement del model.

- PRO: Agent conservatiu, agafa rutes segures.
- CONTRA: Aprèn una política que no és la més òptima.

2.3. Virtualització de xarxes

La virtualització de xarxa (NV) [6] fa referència a l'abstracció de recursos de la mateixa xarxa. Aquesta pot combinar diverses xarxes físiques en una xarxa virtual basada en software o pot dividir una xarxa física en xarxes virtuals separades i independents.

El software de virtualització de xarxa permet als administradors de xarxa moure màquines virtuals a diferents dominis sense tornar a configurar-la. A més, aquest software crea una superposició de xarxa que en pot executar capes virtuals separades a sobre de la mateixa xarxa física.

2.4. Allotjament de xarxes virtuals

L'allotjament de xarxes virtuals [7], [8]consisteix en assignar de manera eficient un conjunt de demandes formades per xarxes virtuals a una xarxa física. Aquestes xarxes estan composades per una sèrie de nodes interconnectats mitjançant links. Els nodes i links tenen demandes, en cas de pertànyer a una xarxa virtual, o recursos, si parlem d'una xarxa física. Aquestes demandes o recursos poden ser CPU, ample de banda, retard, etc.

En complexitat computacional, el problema de VNE està inclòs al grup de problemes NP-difícil [9], ja que aquest vol aconseguir l'allotjament òptim dels recursos per a cada xarxa virtual. Pertànyer al grup NP-difícil significa que, com a mínim, el problema és més difícil que qualsevol dels de tipus NP[10]. Aquests últims es poden resoldre amb una màquina de Turing no determinista emprant una quantitat de temps de computació polinòmica.

Quan s'estudia una VNR i es vol fer l'allotjament d'aquesta VN, es duu a terme en dos escenaris diferents: El que s'encarrega de l'allotjament dels nodes virtuals i el que s'encarrega dels links virtuals.

- **Escenari d'allotjament de node virtual:** Cada node virtual d'una mateixa xarxa ha de ser assignat a un node físic diferent. Per tant, aquesta assignació segueix la següent funció: M_N (): $N^V \rightarrow N^S$. On N^V es refereix al node virtual i N^S al node físic. M_N (M) $\in N^S$. M_N (M) = M_N (N), només si M=N, sempre que sigui possible satisfer les demandes dels nodes virtuals amb els recursos de la xarxa física. Les demandes que poden arribar poden ser: CPU o espai per emmagatzemar, capacitat del node, etc...

- **Escenari d'allotjament de link virtual:** En aquest escenari cada link virtual pot ser allotjat en un sol link físic o en més d'un. L'allotjament de link virtual s'implementa seguint la següent funció: M_L (): $L^V \rightarrow P^S$ per a tots els links virtuals. P^S (M_N (M), M_N (N)) on P^S representa el camí que comença amb el node M_N (M) fins al node M_N (N). Per tant s'ha de complir: M_L (M,N) $\subseteq$ P^S (M_N (M), M_N (N)), sempre que es compleixin les demandes dels links virtuals. Les demandes que es fan en una VNR referents a aquest escenari solen ser de capacitat de link i de retard en la propagació de link.

Per exemple, a la figura 2, hi ha un demostració molt útil d'un escenari en el que es vol fer l'allotjament de dues VNR sobre una xarxa física. En aquest cas, no s'han tingut en compte els recursos de les xarxes, com podien haver sigut pels nodes la CPU o per als links l'ample de banda.

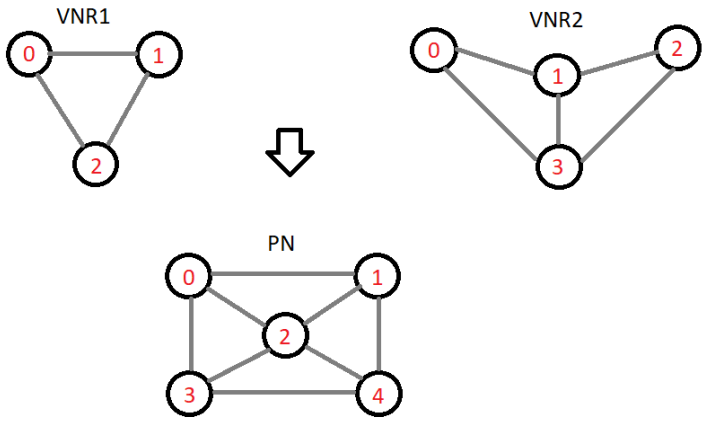

Figura 2: Exemplificació d'un escenari de VNE

Els dos rols que composen el negoci de la virtualització de xarxa dintre el VNE són: proveïdors de servei (SPs) i proveïdors d'infrastuctura (InPs).

- **SPs**: S'encarreguen de proveir i actualitzar els diferents serveis extrem a extrem als usuaris llogant recursos subjacents de diferents InPs.

- **InPs:** S'encarreguen de construir, mantenir i llogar part de les seves xarxes físiques als diferents SPs.

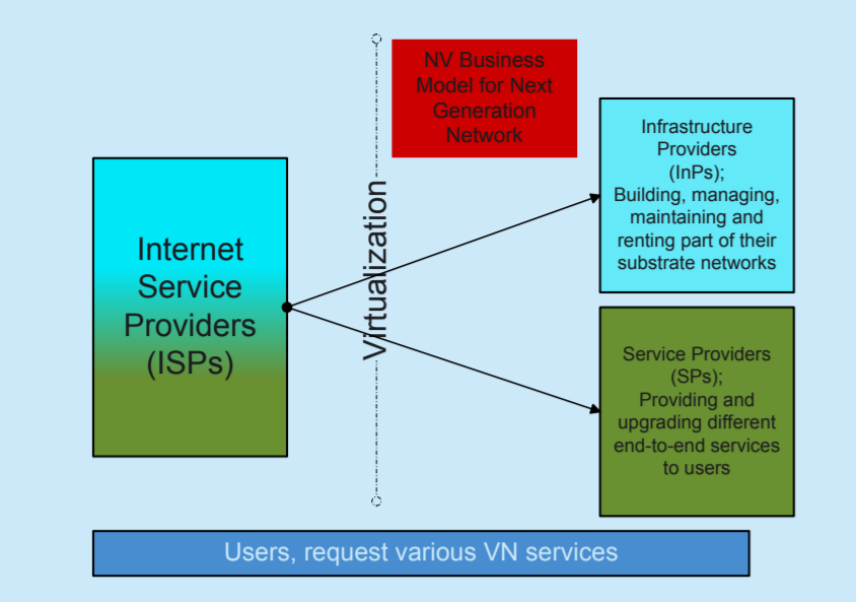

Figura 3: Model de negoci de virtualització de xarxa per a VNE [8]

Respecte al problema de VNE, existia una necessitat de maximitzar els ingressos provinents dels usuaris i de millorar la utilització dels recursos de les xarxes subjacents. Com a solució es va determinar que el millor era idear un algorisme capaç de resoldre la problemàtica.

Els algorismes emprats en la solució es poden classificar de la següent manera:

- **Els exactes:** Algorismes allotgen la VN en cada VNR fent servir una estratègia d'optimització exacta: Programació lineal d'enters (ILP) o programació mixta d'enters. En aquesta estratègia es formula el problema VNE i eines de software s'encarreguen de solucionar el model ILP o MIP formulat. Aquests tipus d'algorismes tenen el handicap de que només són capaços de solucionar petites instancies de problemes VNE. Els temps d'execució a l'allotjament són molt grans.

- **Els heurístics**: No tenen com a objectiu la solució òptima al problema sinó que tracten de trobar-ne una de bona amb un temps d'execució raonable. Aquests poden tenir com a handicap el fet de que poden quedar-se enganxats en un òptim local que queda molt lluny de l'òptim global.

- **Els meta-heurístics**: Volen aconseguir un compromís, quedant a prop de l'allotjament òptim per a un temps d'execució curt. Tot i així, aquest tipus d'algorismes no són els més utilitzats

A la taula 2.2 es poden veure les mètriques més importants en la valoració del rendiment d'un algorisme que pugui resoldre el problema de VNE.

Mètriques d'InP i SP	Relació d'acceptació de VNRs	Mesura el nombre d'allotjaments de VNs satisfactoris amb l'algorisme emprat dividit per el total de propostes de VNs
	El cost d'allotjament	Es refereix al cost total dels elements físics consumits fets servir a l'allotjament. Per exemple: CPU o capacitat de link
	El revenue de l'allotjament	Fa referència a la suma dels recursos dels nodes i links virtuals de les VNs allotjades
	El revenue de l'allotjament respecte el cost	Respecte al rati entre el revenue de l'allotjament i els costos
	Utilització de recursos físics	El rati entre els recursos físics utilitzats entre els totals
	Pena física	Per preparar-se per a una interrupció es defineix aquesta mètrica. Es determina per la suma de tots els elements virtuals afectats, com poden ser el node i els seus links adjacents
	El temps d'execució de l'allotjament	El temps d'execució per un allotjament satisfactori d'una VN
	El temps per tornar a allotjar un VN erràtic	Mostra l'habilitat de l'algorisme per tornar a allotjar una VN quan la VN allotjada ha estat errada
	Número de nodes físics actius	Es fa servir per fer una aproximació en brut del cost d'energia
Mètriques utilitzades per usuaris finals	Llargada del camí d'allotjament	Es refereix al número de links entre dos nodes físics que son allotjats en dos nodes virtuals
	Retard produït a la propagació de link	Mesura la quantitat de temps que es necessita per moure un paquet de dades d'un node físic a un altre

Taula 2: Principals mètriques per a la valoració d'algorismes [8]

2.5. ALEVIN

ALEVIN (ALgorithms for Embedding VIrtual Networks) [11] és un software modular proposat al projecte VNREAL [12] on s'implementen els principals algorismes proposats per a resoldre el problema de VNE. Això permet la comparació entre ells gràcies a la gran quantitat de mètriques que conté. Aquest també compta amb una bona interfície en la que la creació d'escenaris per a l'emulació del problema de VNE és molt flexible i intuïtiva.

3. Recerca

3.1. Anàlisi IA aplicada al problema de VNE

Un dels majors problemes a resoldre dintre de la virtualització de xarxes és el de poder allotjar les demandes de les diferents xarxes virtuals en una xarxa física de la forma més eficient possible, incrementant la utilització de la mateixa i els ingressos dels proveïdors d'infraestructura. Aquest problema es coneix com VNE (virtual network embedding) format per dos grans escenaris: el mapeig dels nodes i el mapeig dels links. Tal i com s'ha exposat en anteriors apartats, diferents algorismes del tipus exacte i heurístic s'han aplicat per solucionar el problema. Els de tipus exacte han aconseguit solucions amb un molt bon ús dels recursos de la xarxa física, però amb un gran temps d'execució, el que no els fa gaire pràctics. D'altra banda, els algorismes heurístics consideren les dues parts del problema de forma separada i a més no poden ser optimitzats per si mateixos, el qual fa que obtinguin solucions no gaire optimes.

Els avenços de la tecnologia als darrers anys han fet possible que molts problemes amb una alta complexitat puguin ser solucionats amb l'ús de la intel·ligència artificial. A fi de poder adreçar una solució més òptima a aquest problema, es vol emular el comportament d'un ésser viu, incrementant el rendiment en la resolució del problema. És per aquest motiu que s'ha considerat la IA com una bona alternativa per a resoldre la problemàtica de VNE.

A continuació es veuran quins camps de la IA poden ser útils per dirimir el problema i quins no.

3.1.1. Aprenentatge supervisat aplicat al problema VNE

En anteriors apartats s'ha vist que els algorismes dintre de l'aprenentatge supervisat tenen dues tasques molt importants, la regressió i la classificació. En aquests algorismes s'entrena amb conjunts de dades que han estat etiquetades, el que l'instrueix sobre quina sortida està més relacionada a cada valor d'entrada. Un cop entrenat l'algorisme, s'entra a una fase de proves on les dades d'entrada estan etiquetades però aquestes no han estat revelades a l'algorisme amb l'objectiu de comprovar si és capaç de rendir bé amb dades no etiquetades.

El problema de VNE es basa principalment en un entorn format per una xarxa física a la que se li fan allotjaments de xarxes virtuals i els recursos d'aquesta van disminuït depenent de la quantitat d'allotjaments i de les respectives demandes. El handicap a l'hora d'escollir aquest tipus d'algorisme per a resoldre el problema és la necessitat de tenir grans conjunts de dades etiquetades llestes per a l'entrenament de l'algorisme. És inviable obtenir aquestes dades d'allotjament amb etiquetes abans d'aplicar cap algorisme, per tant els algorismes pertinents al grup d'aprenentatge supervisat estan descartats com a possible solució del problema.

3.1.2. Aprenentatge no supervisat aplicat al problema VNE

En anteriors apartats s'ha vist que els algorismes dintre de l'aprenentatge no supervisat s'encarreguen de problemes basats en clustering o associació, una de les principals diferències respecte als anteriors tipus d'algorismes és que aquests reben conjunts de dades no etiquetades com a part de l'entrenament, el que fa que no ni hagin valors correctes de sortida i permeten a l'algorisme determinar les similituds i patrons que segueixen les dades. En resum, aquests algorismes són capaços de funcionar lliurement per tal d'aprendre més de les dades i trobar resultats interesants.

De la mateixa manera i així com s'ha exposat en altres apartats, els principals algorismes pertinents a aquest grups s'utilitzen en l'agrupació de les dades en grups mitjançant les seves característiques. Un exemple d'ús podria ser les recomanacions que fa una web per veure pel·lícules en streaming: Quan l'usuari acaba de visualitzar una pel·lícula, la web sol recomanar tot un llistat de contingut semblant.

Aquest tipus d'aprenentatge tampoc és el més òptim per aplicar al problema de VNE, el que fa que quedi descartat, ja que no es vol resoldre cap problema de clustering o agrupacions. A banda d'això, també es necessitaria de grans conjunts de dades d'allotjament per a l'entrenament.

3.1.3. Aprenentatge de reforç aplicat al problema VNE

Ja hem parlat de que els algorismes pertinents al tipus d'aprenentatge de reforç treballen en problemes d'explotació o exploració, on hi ha un entorn semblant al que es pot veure a la figura 1. L'algorisme aprèn a reaccionar a l'entorn per si sol, a prendre decisions en cada estat i a rebre recompenses depenent de la qualitat de l'acció en el mateix estat.

Quan s'enfoca aquest tipus d'aprenentatge en el problema de VNE es pot veure que els algorismes d'aquest grup son vàlids ja que necessiten d'un entorn per a poder treballar com el que podria ser l'escenari d'una simulació de xarxa física. Un altre punt a favor d'aquest tipus d'aprenentatge és que no necessita cap tipus de dades d'entrada, ja que aprèn amb un mecanisme d'estat-acció rebent un feedback del propi entorn.

Degut als motius esmentats, aquest tipus d'aprenentatge serà l'escollit perquè es tracta del més òptim entre els estudiats.

A continuació s'estudiaran els principals algorismes que existeixen per tal de trobar el més útil a la resolució del problema de VNE.

3.1.1.1. Algorisme Sarsa

Per tal d'aconseguir la funció acumulativa de descompte màxima, la Q òptima d'estat-acció ha de satisfer la següent formula:

$$Q^*(s,a) = \sum_{s \in S} P^a_{ss'}[R^a_{ss'} + \gamma max_{a \in A} Q^*(s',a)]$$

Aquest algorisme selecciona el valor Q en cada iteració a partir de l'experiència (s_t,r_t,s_{t+1}), llavors en cada pas l'algorisme es pot definir com:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Figura 4: Algorisme SARSA [13]

3.1.1.2. Algorisme Q learning

En aquest algorisme cada estat-acció correspon a un valor Q i s'escull cada acció depenent del valor Q. El valor Q es pot obtenir de la següent formula:

$$Q(s,a) = R(s,a) + \gamma \sum_{s \in S} P^a_{ss'} V(s')$$

Si tenim que:

$$V(s') = max_{a \in A} Q(s,a)$$

Substituint:

$$Q(s,a) = R(s,a) + \gamma \sum_{s \in S} P^a_{ss'} max_{a \in A} Q(s',a')$$

On $R(s,a)$ és la recompensa instantània al dur a terme una acció sota un estat s.
Per afegir diferència temporal a aquest algorisme a fi de tenir en compte els canvis a l'entorn durant el temps, es pot emprar la següent equació:

$$TD(s,a) = R(s,a) + \gamma \sum_{s \in S} \left(P^a_{ss'} max_{a'} Q(s',a')\right) - Q(s,a)$$

En conclusió, els nous valors de Q s'actualitzen de la següent forma:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha TD_t(a,s)$$

On α és el rati d'aprenentatge que controla com de ràpid apren l'agent els canvis aleatoris imposats per l'entorn, $Q_t(s,a)$ és el valor actual de Q i $Q_{t-1}(s,a)$ és el valor anterior de Q.

Per tant, si canviem $TD_t(a,s)$ per la seva expressió sencera, l'equació que obté els valors de Q i resol la diferencia temporal és la següent:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma max_{a'} Q(s',a')) - Q_{t-1}(s,a))$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state,\cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$
  until S is terminal

Figura 5: Algorisme Q-Learning [13]

3.1.1.3. Algorisme aprenentatge reforç escollit

Un cop analitzats els principals algorismes que existeixen en el tipus d'aprenentatge de reforç, s'ha d'escollir quin serà l'encarregat de solucionar el problema de VNE.

D'acord amb el que s'ha tractat anteriorment, l'algorisme de SARSA està basat en model, el que vol dir un cop escollim una acció a_t i la següent acció a_{t+1} en un estat s, un cop es canvia al següent estat s_{t+1} l'acció escollida serà a_{t+1}. És a dir, es manté la política. Per contra, a l'algorisme Q learning no està basat en model sinó que quan s'actualitza el valor de $Q(S_t,A_t)$ s'agafa l'acció que obtingui el valor $Q(S_{t+1},A_{t+1})$ més gran, però un cop arribat a l'estat s_{t+1}, hi ha una probabilitat de no escollir l'acció a_{t+1}, derivada del model escollit com pot ser el model Epsilon-Greedy algorithm.

L'algorisme de SARSA permet arribar al destí seguint un camí sense riscos, el que el converteix en una bona opció si al problema al que s'ha d'aplicar penalitza molt el fet d'escollir accions errònies. Normalment aquest algorisme sol trigar més en trobar una solució al problema.

L'algorisme Q learning arriba al destí o al final del problema seguint un camí a base d'assolir i assumir riscos. No es tracta d'una bona opció si el problema penalitza molt el fet d'agafar

accions errònies. No obstant, aquest algorisme sol trigar menys en trobar una solució que l'anterior.

Un cop exposades les propietats, els avantatges i desavantatges de cada algorisme, per a solucionar el problema de VNE s'ha escollit Q learning, ja que en aquest tipus de problema es mira molt el temps d'execució i el fet de prendre decisions errònies no és un problema perquè el problema s'ha de resoldre de manera offline. Això vol dir que fins que l'algorisme no trobi una solució correcta sobre un allotjament, no anirà a fer el següent de manera que la pressa de decisions i el fet d'equivocar-se a l'hora de trobar el millor allotjament no serà un punt crucial.

3.2. ALEVIN software

Un cop escollit l'algorisme amb el que es tractarà el problema de VNE, es crucial trobar un entorn on desenvolupar, implementar l'algorisme i comparar els resultats amb altres algorismes. ALEVIN (figura 6) codificat amb Java, gestiona diversos tipus d'algorismes offline i paràmetres arbitraris de recursos i demandes. Amb l'ajuda del seu manual d'usuari inclòs al programari a la descarrega [14], habilita als desenvolupadors o investigadors a poder afegir els algorismes, paràmetres o mètriques pròpies al software. Aquest presenta una interfície gràfica que permet visualitzar les xarxes físiques i virtuals com grafs dirigits per tal de facilitar que les simulacions siguin més intuïtives.

A aquest fi, s'ha escollit ALEVIN com a entorn per desenvolupar, implementar i comparar l'algorisme, en els següents apartats, es presentaran aspectes importants del simulador, com l'entorn en el que s'ha instal·lat aquest software, els passos que s'han de seguir per implementar un nou algorisme o com és dur a terme l'avaluació d'un algorisme.

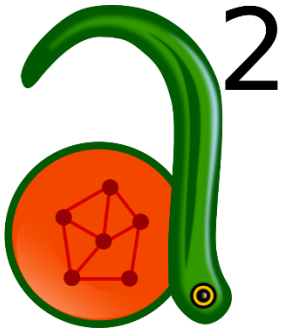

Figura 6: ALEVIN logo [14]

3.2.1. Entorn de simulació

L'entorn recomanat per executar ALEVIN és Linux. En el cas que ateny a aquest treball s'ha escollit una imatge d'Ubuntu, que es pot aconseguit a qualsevol sofware de virtualització com podria ser VirtualBox[15]. Els passos que s'han de seguir per dur a terme una imatge d'Ubuntu són els següents: Prémer al botó 'Nova' (figura 7) i escollir el tipus de sistema operatiu i versió que es vol instal·lar la imatge. Tot seguit, s'han de triar les característiques com memòria RAM o mida de disc dur i muntar la imatge. Un cop instal·lada, la imatge s'obre fent un doble clic. Després d'esperar un cert temps de càrrega, apareix el sistema operatiu d'Ubuntu que s'ha creatyu sobre VirtualBox.

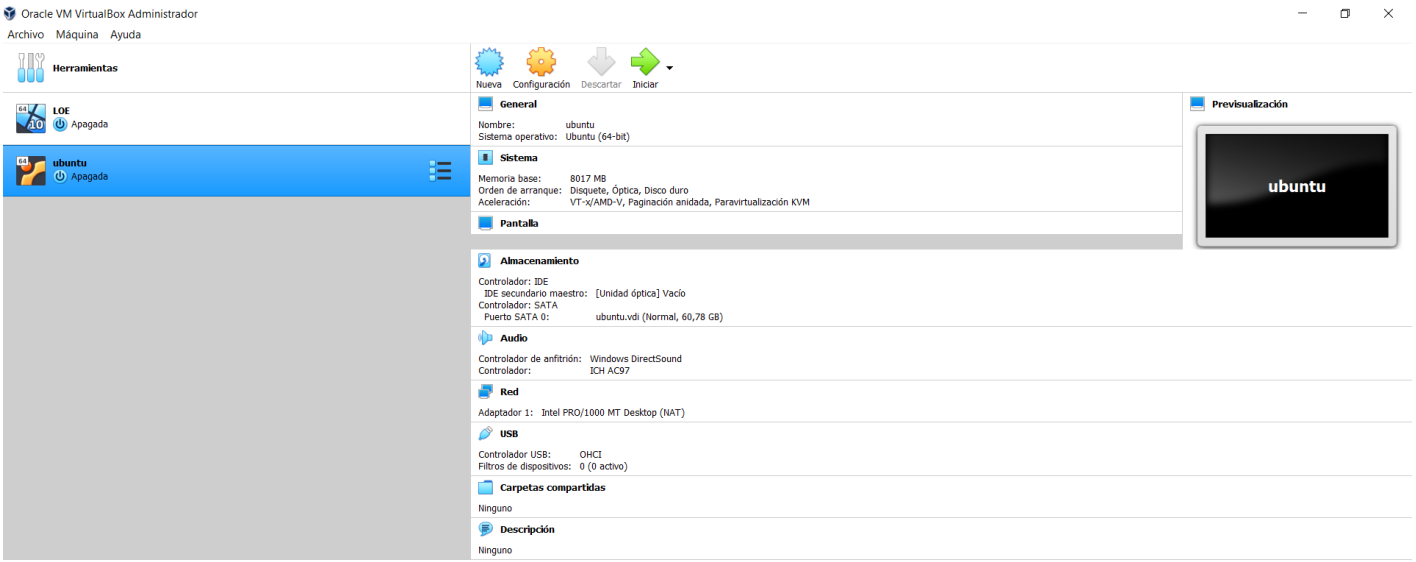

Figura 7: Interficie VirtualBox

El primer pas que s'ha de seguir és el de compilació, per això des de la carpeta que conté el projecte d'ALEVIN, s'ha d'obrir un terminal i executar l'script ANT per crear els executables necessaris per al projecte. El segon pas és el d'execució del programa i es pot fer de dues maneres: La primera seria mitjançant un terminar i executant la comanda "java -jar ALEVIN.jar" i la segona seria obrint Eclipse[16], un Entorn de Desenvolupament Integrat (IDE, segons les seves sigles en anglès) que s'utilitza per desenvolupar projectes. Un cop obert només s'ha d'executar la següent classe "vnreal.Main". Un cop executada alguna d'aquestes opcions, s'hauria d'obrir la interfície gràfica del programa.

3.2.2. Interfície gràfica

La interfície gràfica d'ALEVIN (figura 8) presenta una forma molt clara, amb tres subdivisions:

- La part central, on es mostren les xarxes de forma gràfica.
- La part de la dreta, on es poden veure els links seleccionats sobre la part gràfica a la pestanya de "selection" i els mapejats de les xarxes virtuals a la xarxa física a la pestanya de "mapping".
- La part inferior, la consola, on principalment es veu si hi ha hagut cap problema a l'hora d'executar un algorisme.

A la part superior de la interfície es mostren els típics menús desplegables, amb els següents elements:

- File: Permet importar i exportar escenaris amb els seus paràmetres, demandes i recursos en diversos formats. També permet crear xarxes físiques o virtuals de forma manual, afegint nodes i links un per un.
- View: Permet veure la ID assignada als links i nodes de les xarxes virtuals i físiques sobre la part central on estan presentades en forma de dibuix.
- Generators: Permet generar xarxes físiques i virtuals de forma automàtica amb la capacitat d'escollir el tipus de generador que es vol fer servir, la quantitat de nodes, links o altres paràmetres relacionats amb la creació de xarxes. També permet afegir els recursos a les xarxes físiques o demandes en cas de xarxa virtual de forma automàtica, de tal manera que s'assignen uns recursos a la xarxa de forma aleatòria entre 0 i la quantitat màxima de recursos que s'han assignat.
- Algorithms: Un cop generades les xarxes amb els seus recursos, aquesta pestanya permet escollir l'algorisme amb el que es vol solucionar el problema i començar la simulació.
- Metrics. Després d'iniciar la simulació amb l'algorisme escollit, aquesta pestanya permet veure el resultat de les diferents mètriques que hi ha per a la seva evaluació.

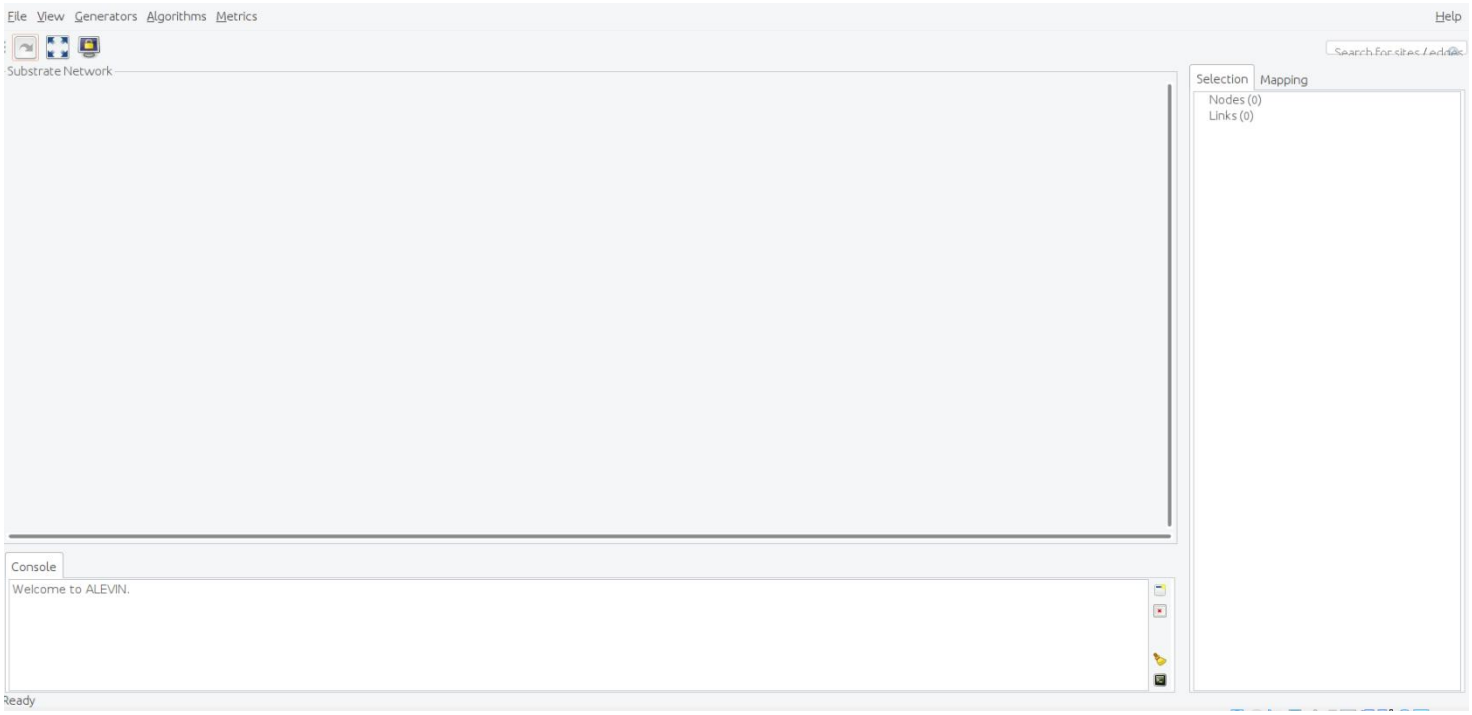

Figura 8: interfície ALEVIN

4. Desenvolupament

4.1. Definició del sistema Q learning per a VNE

4.1.1. Entorn

La definició de l'entorn és clau a l'hora d'implementar l'algorisme per a resoldre el problema de VNE. L'entorn consta d'una xarxa física amb diferents recursos en links i nodes, formats per ample de banda i CPU respectivament, i de diverses sol·licituds en forma d'allotjaments de xarxes virtuals amb diferents demandes de recursos, composat per ample de banda i CPU en links i nodes respectivament.

En anteriors apartats s'ha parlat sobre com l'estructura dels algorismes d'aprenentatge de reforç (figura 1) consta bàsicament d'un agent i un entorn i entre aquests dos intercanvien informació mitjançant estats, accions i recompenses. L'agent prem una acció, l'entorn genera una recompensa depenent de l'acció escollida i retorna un nou estat a l'agent. Tenint en compte això, es determina que l'estratègia més òptima a dur a terme és que per cada xarxa

virtual que es vulgui allotjar s'haurà d'inicialitzar un agent que tindrà accés als recursos de la xarxa física i de la xarxa virtual en qüestió.

Les accions que prendrà un agent hauran de fer referencia als nodes de la xarxa física, de tal manera que per cada estat l'agent tindrà un conjunt d'accions: $A = \{0 \dots \dots \dots N^f\}$, on N^f equivaldrà a la quantitat total de nodes de la xarxa física que es podran escollir per fer l'allotjament en el mateix estat. En definir les accions com al conjunt de nodes de la xarxa física, els estats seran els nodes de la xarxa virtual en qüestió que es voldran allotjar i, per tant, l'agent també tindrà un conjunt d'estats $S = \{0 \dots \dots \dots N^v\}$, on N^v farà referència a la quantitat total de nodes de la xarxa virtual que formaran els estats.

Amb les definicions dels estats i accions, els allotjaments dels nodes estan coberts, de tal manera que un dels possibles allotjaments de la xarxa virtual 1 de la figura 2 s'escriuria amb la següent notació: {0,0}, {1,1}, {2,2}. On les tuples estan formades per parells d'estats accions $\{S, A\}$.

4.1.2. Matriu Q

En aquest algorisme, la construcció d'una matriu Q on es guardin els resultats de la qualificació Q de cada parell estat-acció és necessari. Per definició, aquesta matriu té com a columnes el conjunt d'accions que pot prendre l'agent i com a files el conjunt d'estats que pot escollir l'agent (figura 9).

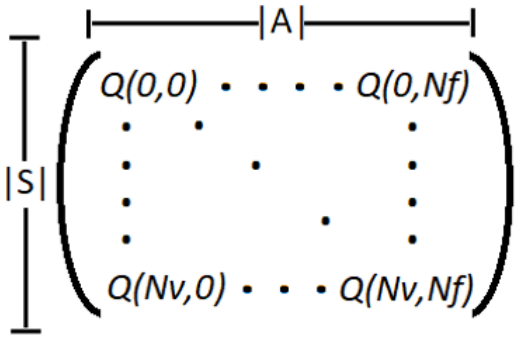

Figura 9: Matriu Q

El número de files d'aquesta matriu és $|S|$ i el d'accions $|A|$. Com ja em aclarit, els estats representen el conjunt de nodes de la xarxa virtual que es volen allotjar i les accions representen el conjunt de nodes de la xarxa física. Per tant, cada element $Q[S_i, A_i]$ representa la qualitat d'aquella parella d'estat-acció, el que vol dir que un valor gran de Q correspondrà amb una bona parella estat-acció i un valor petit de Q correspondrà amb una parella dolenta estat-acció.

Al començament de l'allotjament d'una xarxa virtual, l'agent inicialitza la matriu Q a zeros. Tal com es veu a la figura 5, en cada iteració l'agent anirà actualitzant aquests valors de Q fins que arribi a un número d'iteracions en que aquesta taula és mantingui estable.

4.1.3. Actualització de valor Q

A cada elecció de parell estat-acció s'ha de calcular el valor que li pertany, segons l'equació que calcula el valor Q :

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma max_{a'} Q(s',a') - Q_{t-1}(s,a))$$

On $Q_{t-1}(s,a)$ és el valor actual que hi ha a la taula corresponent a l'elecció estat-acció, α és el valor d'aprenentatge que controla com de ràpid l'agent adopta els canvis aleatoris imposats per l'entorn, $R(s,a)$ és la recompensa proveïda a l'agent per agafar el parell d'estat-acció corresponent, γ és un factor de descompte corresponent a l'afectament de futurs estats a l'actual, $max_{a'} Q(s',a')$ és el màxim valor Q corresponent al nou estat.

Tal com s'ha plantejat l'algorisme en el problema, l'elecció dels estats feta per l'entorn no depèn de l'acció escollida en l'anterior estat, ja que no té lògica escollir el següent node virtual a allotjar a partir de l'acció seleccionada a l'anterior estat. Aquesta peculiaritat fa que la dependència entre estats sigui nul·la, obligant a dur a terme una sèrie d'adaptacions a l'equació que calcula el valor de Q en cada estat esborrant el factor que calcula la incidència de futurs parells estat-acció a l'actual:

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) - Q_{t-1}(s,a)) \qquad (1)$$

Amb aquesta equació es pretén que l'algorisme explori l'entorn i aprengui estratègies, considerant l'experiència o valors anteriors independentment dels estats futurs.

4.1.4. Funció de recompensa

Al problema de VNE s'ha de tenir en compte dos escenaris: el de mapeig de nodes i el de links. Ja s'ha parlat sobre com es definien els estats com nodes virtuals i les accions com nodes físics, deixant enllestit l'escenari de mapeig de nodes.

L'escenari de mapeig de links comporta una dificultat més gran que el mapeig de nodes, ja que un link virtual pot ser allotjat en un camí físic que pot comprendre més d'un link. El que fa que un link físic pugui rebre allotjaments de diferents links virtuals, sempre que sigui capaç de satisfer les demandes depenent dels seus recursos, contràriament dels nodes físics, que només poden allotjar un node virtual cada un. Per obtenir una correlació entre el mapeig de nodes i el mapeig de links, s'ha decidit implementar una funció de recompensa que tingui en compte el mapeig dels links. A cada elecció d'allotjament d'un node, aquesta funció de recompensa trobarà els links virtuals entre els nodes virtuals ja allotjats i el del estat actual i farà el mapeig dels mateixos sobre els camins físics pertinents.

Per obtenir una millor optimització dels recursos, els camins de la xarxa física on han de ser allotjats els links virtuals seran els camins més curts. És a dir s'emprarà un algorisme que trobarà els camins més curts de la xarxa i es veuran si aquests són aptes per allotjar depenent si poden satisfer les demandes de recursos o no.

Els camins més curts es trobaran arrel de l'ús de l'algorisme de Dijkstra. Aquest, un cop donat un parell de nodes, et retorna el camí més curt, present al software ALEVIN gràcies a la llibreria JUNG2 [17].

La funció de recompensa pot ser definida de la següent manera:

$$R(s,a) = \sum_{i=0}^{N^{Ve}} \frac{1}{S_i} M(link_i) \qquad (2)$$

On N^{Ve} són el número de nodes que han estat allotjats als estats previs a l'actual, S_i el número de salts que hi ha al camí físic on es vol allotjar el $link_i$ de tal manera que es penalitzin els camins escollits amb molts salts i $M(link_i)$ retorna un valor segons el mapeig del link $link_i$ amb l'algorisme Dijkstra. Podrà ser un número alt si s'ha pogut fer el mapeig o un número baix si no s'ha pogut fer per culpa de les restriccions marcades pels recursos actuals.

Amb la intenció d'assolir un allotjament més òptim i una convergència més ràpida, s'ha afegit una funció que a l'últim estat de cada allotjament en cada iteració s'actualitzen els valors Q corresponents a l'elecció de cada estat-acció. Així, si l'allotjament de la xarxa virtual ha estat un èxit, es suma una constant a cada valor Q escollit, però si no s'ha aconseguit fer l'allotjament, es resta una constant a cada valor Q escollit.

4.1.5. Compromís exploració-explotació

Per obtenir una matriu Q estable, on totes les combinacions d'estat-accions s'hagin vist, cal seguir una regla d'exploració-explotació. L'agent, al començament de l'entrenament, no compta amb informació i tots els seus valors Q són zeros, per tant es requereix d'un model capaç de balancejar l'ús de l'exploració i explotació en l'algorisme, afavorint l'exploració en les primeres iteracions per descobrir nous valors Q i l'explotació en les darreres triant els millors valors Q. Per això, es segueix el model epsilon Greedy, en el que profunditzarem al següent apartat.

4.1.6. Model Epsilon-Greedy

Aquest model és un mètode senzill per escollir entre exploració i explotació de forma aleatòria, amb probabilitat ε per exploració i 1- ε per explotació. Per tal d'obtenir uns valors més estables de Q i una millor convergència, s'ha decidit donar a ε valors dinàmics, de tal manera que vagi decreixent a mesura que l'algorisme va sumant iteracions. Això fa que pràcticament s'obligui a l'algorisme a explorar al principi quan la taula Q està buida i a explotar al final quan s'han descobert totes les combinacions possibles. L'equació que calcula ε en cada iteració és la següent:

$$\varepsilon = \min_{\varepsilon} + \frac{1 - \min_{\varepsilon}}{e^{-\lambda * \mathrm{num}_{iter}}}$$

On $\min_{\varepsilon}$ és ε mínima per tal de que la funció mai sigui 0, λ un factor que serveix per atenuar la funció i que trigui en decreixer més iteracions i num_{iter} és el número d'iteracions al moment actual. D'aquesta manera, per cada iteració el valor de epsilon anirà decreixent amb la

intenció d'anar prioritzant l'explotació a futures iteracions. A la figura 10 es pot veure els valors que compren èpsilon en el transcurs del entrenament.

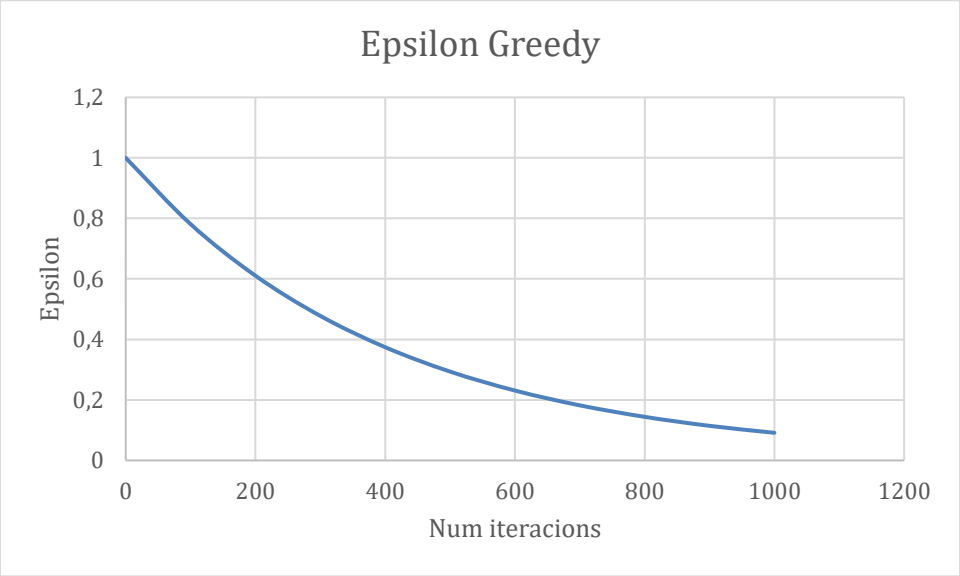

Figura 10: Gràfica valors d'èpsilon

4.1.7. Pseudocodi algorisme aplicat a VNE

La idea principal d'aquest algorisme és la d'actualitzar contínuament la matriu Q amb els valors Q obtinguts de l'elecció estat-acció i les seves corresponents recompenses. A la llarga, després d'un cert numero d'iteracions, la matriu Q comença a obtenir una bona estabilitat i això deriva en un allotjament de la xarxa virtual pràcticament òptim.

L'algorisme consta d'un bucle principal que farà repetir el codi, el número d'iteracions que es marqui amb la variable max_iter. Dintre d'aquest bucle, un altre bucle for itera sobre tots els estats, sobre els nodes de la xarxa virtual. Per cada node virtual si ε és més petit que num_random, escollirà l'acció amb millor valor Q en aquell estat, sempre que compleixi amb les limitacions imposades als nodes. Si ε és més gran que num_random, l'algorisme prendrà una acció aleatòria de les possibles en aquell estat. Un cop decidida una acció en l'estat actual, es calcula la recompensa per escollir aquella acció o node físic i aquesta recompensa depèn del mapeig dels links virtuals que lliguen aquell node virtual amb els nodes virtuals ja allotjats. Un cop calculada la recompensa, actualitza el valor Q amb la formula d'actualització de valor Q i guarda aquest valor Q a la matriu.

Un cop el for arriba a l'últim estat o l'últim node virtual en ser allotjat, després d'actualitzar la taula Q amb el valor Q corresponent a l'elecció de l'acció, s'entra a l'últim if, ja que compleix la condició. Dintre d'aquest if es recopilen tots els parells estat-acció escollits en aquella iteració: si l'allotjament ha estat satisfactori, s'actualitzen els valors Q corresponents amb un valor superior al que tenien abans, sumant-li una constant i en cas contrari s'actualitzen amb un valor inferior al que tenien abans restant-li la mateixa constant.

Algorisme Q learning basat en VNE

```
1  while ( num_iter < max_iter){
2     for(i=0;  i< N^v;  i++){
3        if(num_random > ε){
4           Escull acció A_i amb millor valor Q per aquest estat
5           Ha de satisfer les limitacions imposades als nodes
6        }else{
7           Prem una acció aleatòria per aquest estat
8        }
9        Calcula la recompensa R com a l'equació (2)
10       Calcula el valor Q com a l'equació (1)
11       Actualitza la taula Q amb el valor Q calculat
12       If(i==N^v){
13          Actualitza tots els valors Q d'aquesta iteració amb una recompensa nova segons si
14    s'ha completat l'allotjament
15       }
16    }
17    num_iter++;
18    ε = min_ε + (1 − min_ε) / e^(−λ∗num_iter)
19    Return de l'allotjament basat en la taula Q
20 }
```

4.2. Implementació de l'algorisme a l'ALEVIN

ALEVIN [14] compta amb un manual d'usuari per als desenvolupadors que vulguin afegir algorismes, paràmetres, recursos o mètriques noves. Si es vol afegir un algorisme nou per resoldre el problema VNE que utilitzi VNREAL, s'ha de derivar de la interfície mulavito.IAlgorithm, aquesta permet:

- Definir una manera comuna per accedir a la informació de l'estat de l'algorisme.
- Permet tenir una barra de progressió que es mostra a la interfície gràfica.

A l'ALEVIN hi ha dues formes d'implementar els algorismes, segons com s'hagi pensat. La primera és implementar el mapeig dels nodes i de links en el mateix escenari i la segona és implementar el mapeig de nodes i de links en escenaris independents, el que requereix més classes.

Tal com s'ha implementat Q learning, aquest fa servir un sol escenari per completar el mapeig de nodes i links. Les classes disponibles de les que s'ha d'estendre, per tal de tenir mètodes bàsics per a la implementació del algorisme, són les següents: AbstractAlgorithm.java, AbstractSequentialAlgorithm.java, AbstractRevokableSequentialAlgorithm.java, GenericMappingAlgorithm.java.

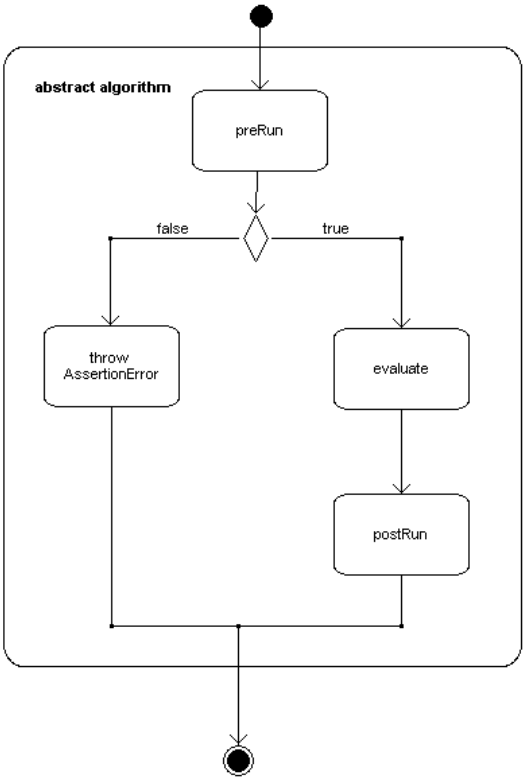

Figura 11: Diagrama de flux d'Algorisme abstracte [14]

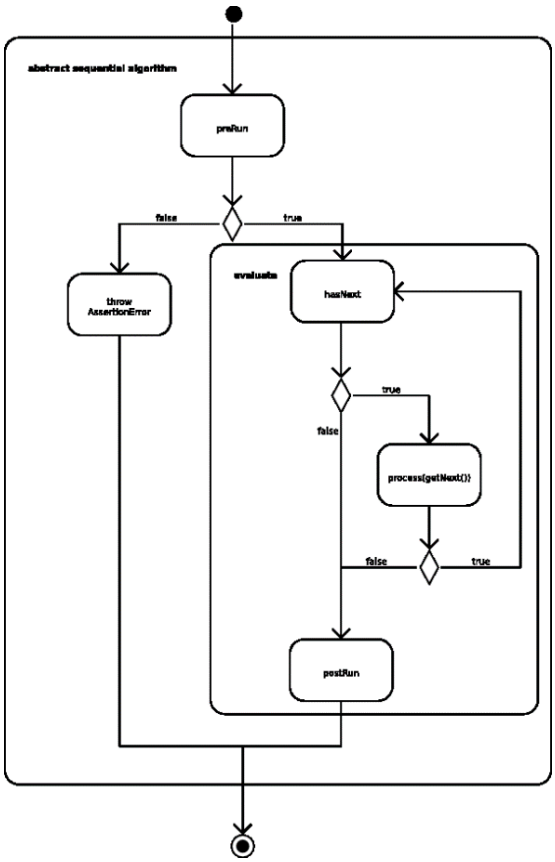

Figura 12: Diagrama de flux d'algorisme abstracte seqüencial [14]

S'ha escollit el tipus d'algorisme abstracte seqüencial com a base, ja que realitza un processament seqüencial proporcionant mètodes abstractes hasNext i getNext.

El principi bàsic dels algorisme d'ALEVIN és el següent:

- Obtenir vnreal.network.NetworkStack que consisteix en:
    - Una xarxa física amb recursos.
    - Una llista de xarxes virtuals amb les seves demandes.
- Fer els allotjaments d'una xarxa virtual buscant els recursos de la xarxa física que poden satisfer les demandes de la xarxa virtual.

4.2.1. implementació dels mètodes heretats de l'algorisme abstracte seqüencial

A l'hora d'implementar els mètodes heretats, s'han d'inicialitzar una sèrie de variables necessàries; una que contingui la xarxa física corresponent, una altre que sigui un iterador entre totes les xarxes virtuals que ni han en forma de demanda i per últim un iterador que contingui la xarxa virtual a la que s'està sometent l'allotjament:

```
1       private final NetworkStack stack;
2       private Iterator<? extends Network<?, ?, ?>> curNetIt;
3       private Iterator<VirtualNetwork> curIt;
```

Estenent de l'algorisme abstracte seqüencial s'hereten uns mètodes necessaris per al desenvolupament de l'algorisme, que són els següents:

- **Mètode recupera estadístiques:** Aquest mètode retorna una llista amb els elements que es volen mostrar en la finalització de l'execució del algorisme. Molt útil per tenir en compte el número de links i nodes dels que s'ha fet el mapeig.

```
1   @Override
2   public List<AbstractAlgorithmStatus> getStati() {
3
4   LinkedList<AbstractAlgorithmStatus> stati = new
5   LinkedList<AbstractAlgorithmStatus>();
6
7         stati.add(new AbstractAlgorithmStatus("Mapped VN nodes") {
8               private int max = -1;
9
10              @Override
11              public Integer getValue() {
12                    return mappedNodes;
13              }
14
15              @Override
16              public Integer getMaximum() {
17                    if (max == -1) {
18                          max = 0;
19                          int size = stack.size();
20                          for (int i = 1 /* skip substrate */; i < size; i++)
21                                max += stack.getLayer(i).getVertexCount();
22                          }
23                          return max;
24                    }
25              });
26        stati.add(new AbstractAlgorithmStatus("Mapped VN links") {
27              private int max = -1;
28
29              @Override
30              public Integer getValue() {
31                    return mappedLinks;
32              }
33
```

```
34              @Override
35              public Integer getMaximum() {
36                    if (max == -1) {
37                          max = 0;
38                          int size = stack.size();
39                          for (int i = 1 /* skip substrate */; i < size; i++)
40                                max += stack.getLayer(i).getEdgeCount();
41                          }
42                          return max;
43                    }
44              });
45        return stati;
46  }
```

- **Mètode process:** Aquest mètode obté com a paràmetre una xarxa virtual de la que s'ha de fer l'allotjament. És el mètode principal on s'ha de desenvolupar l'allotjament d'una xarxa virtual, on s'obtenen les dades de la xarxa virtual, es crea la matriu Q amb les mides de la xarxa física i xarxa virtual corresponent i es realitza l'allotjament i retorna un booleà. Si retorna false, l'algorisme para la seva execució tot i que hi hagi més xarxes virtuals per allotjar.

```
47  @
1   Override
2         protected boolean process(VirtualNetwork p) {
3           actions=getActions();
4           states=getStates(p);
5           matriuQ = new double [states.size()][actions.size()];
6           getBestMapping(p,training(p));
7           return true;
8         }
```

- **Mètode preRun i postRun:** Aquests mètodes tenen en compte el temps en el que comença i finalitza l'algorisme.

```
1   @Override
2         protected boolean preRun() {
3               startTime=System.currentTimeMillis();
4               return true;
5         }
6
7         @Override
8         protected void postRun() {
9               // TODO Auto-generated method stub
10              runningTime = (System.currentTimeMillis() - startTime) / 1000;
11
12        }
```

- **Mètode hasNext i getNext:** El primer mètode actualitza l'iterador de les xarxes virtuals, si una demanda de xarxa virtual ha estat feta mou l'iterador a la següent xarxa virtual. El segon mètode retorna la següent xarxa virtual que serà processada en el mètode process.

```
1  @Override
2        protected boolean hasNext() {
3              if (curNetIt == null) {
4                    curNetIt = stack.iterator();
5              }
6              if (curIt == null || !curIt.hasNext()) {
7                    if (curNetIt.hasNext()) {
8                          @SuppressWarnings("unused")
9                          Network<?, ?, ?> tmp = curNetIt.next();
10                         curIt = (Iterator<VirtualNetwork>) curNetIt;
11                         return hasNext();
12                   } else
13                         return false;
14             } else
15                   return true;      }
16
17       @Override
18       protected VirtualNetwork getNext() {
19             if(!hasNext()) {
20                   return null;
21             }else {
22                   return curIt.next();
23                   }     }
```

4.2.2. Implementació mètodes no heretats de l'algorisme abstracte seqüencial

```
1        protected List<VirtualNode> getStates(VirtualNetwork p) {
2              List<VirtualNode> states = new LinkedList<VirtualNode>();
3              for(VirtualNode aux : p.getVertices()) {
4                    states.add(aux);
5              }
6              return states;
7        }
```

```
1  protected List<SubstrateNode> getActions() {
2          List<SubstrateNode> actions = new LinkedList<SubstrateNode>();
3          for(SubstrateNode aux : stack.getSubstrate().getVertices()) {
4              actions.add(aux);
5          }
6          return actions;
7      }
```

```
protected boolean fulfillDemand(VirtualNode a, SubstrateNode b) {
            boolean fulfill=true;
            for(AbstractDemand dem: a) {
                  for(AbstractResource res: b) {
                        if(res.fulfills(dem) && res.accepts(dem)) {
                              fulfill=true;
                        }else {
                              return fulfill=false;
                        }
                  }
            }
            return fulfill;
      }
```

```
protected double epsilonCalculator(int numIter) {
      double factorDescompte=0.001;
      double epsilonMinValue=0.01;
      return epsilonMinValue+((1 - epsilonMinValue)/Math.pow(Math.E,
factorDescompte*numIter));
      }
```

```
1   private List<Integer> training(VirtualNetwork p) {
2   int numIter=0;
3   double maxValorActState=0;
4   HashMap <VirtualNode, SubstrateNode> mappedNodes = new HashMap <VirtualNode, SubstrateNode> ();
5   Random r= new Random();
6   int numAction=0;
7   double qValue=0;
8   List<SubstrateNode> actionsAvailable ;
9   List<Integer>numsActionsChoosen;
10
11  while (numIter<maxIter){
12        double reward=0;
13        double rewardFirstState=0;
14        actionsAvailable=new ArrayList<SubstrateNode>(actions);
15        boolean PossibleActions=true;
16        numsActionsChoosen=new ArrayList<Integer>();
17        double num_random=Math.random();
18
19        for(int i=0; i< states.size(); i++) {
20
21              if(num_random > epsilon ){
22                    maxValorActState=matriuQ[i][actions.indexOf(actionsAvailable.get(0))];
23                    for(int j=0; j<actionsAvailable.size(); j++) {
24                          if(maxValorActState<=matriuQ[i][actions.indexOf(actionsAvailable.get(j))]
25                          && fulfillDemand(states.get(i),actionsAvailable.get(j))) {
26                                numAction=actions.indexOf(actionsAvailable.get(j));
27                                maxValorActState=matriuQ[i][actions.indexOf(actionsAvailable.get(j))];
28                    }
29                    }
30              }else {
31                    numAction=actions.indexOf(
32                                              actionsAvailable.get(r.nextInt(actionsAvailable.size()))
33                                              );
34
35              }
36
37              int sign;
38              reward=mapLinksandReward(states.get(i),actions.get(numAction), mappedNodes,p,false);
39              qValue=((1-learningRate)*matriuQ[i][numAction])+(learningRate*(reward));
```

```
40                      mappedNodes.put(states.get(i), actions.get(numAction));
41                      matriuQ[i][numAction]=qValue;
42                      numsActionsChoosen.add(numAction);
43                      actionsAvailable.remove(actions.get(numAction));
44                      if(p.findEdge(states.get(0), states.get(i))!=null ||
45                      p.findEdge(states.get(i), states.get(0))!=null) {
46                            rewardFirstState=rewardFirstState+reward;
47                      }
48                      if(i==(states.size()-1)){
49                            matriuQ[0][numsActionsChoosen.get(0)]=(1-learningRate)*
50                            matriuQ[0][numsActionsChoosen.get(0)]+(learningRate*(rewardFirstState));
51                            if(processedLinks==p.getEdgeCount()) {
52                                  sign =1;
53                            }
54                            else {
55                                  sign =-1;
56                            }
57                            for(int l=0; l<states.size(); l++) {
58
59      matriuQ[l][numsActionsChoosen.get(l)]=matriuQ[l][numsActionsChoosen.get(l)]+(sign*10000);
60
61                            }
62                      }
63                }
64                p.clearVnrMappings();
65                processedLinks=0;
66                mappedNodes.clear();
67                numIter++;
68                epsilon=epsilonCalculator(numIter);
69          }
70          if(numIter ==maxIter-1){
71      numsActionsChoosen.clear();
72      for(int a=0; a< states.size(); a++){
73            double auxFirstValue=matriuQ[a][0];
74            int numAction=0;
75            for(int b=1; b< actions.size(); b++){
76                  if(matriuQ[a][b]> auxFirstValue){
77                        auxFirstValue=matriuQ[a][b]
78                        numAction=b;
```

```
79                      }
80                }
81                numsActionsChoosen.add(b);
82          }
83    }
84
85                return numsActionsChoosen;
86          }
```

```
1   private boolean nodeEmbedding(VirtualNode vn, SubstrateNode sn) {
2               boolean fulfilled = false;
3               for (AbstractDemand dem : vn) {
4                     fulfilled = false;
5                     for (AbstractResource res : sn)
6                           if (res.accepts(dem) && res.fulfills(dem) && dem.occupy(res)) {
7                                 fulfilled = true;
8                                 break;
9                           }
10                          else {
11                                return fulfilled=false;
12                          }
13              }
14              if(fulfilled)
15              mappedNodes++;
16              return fulfilled;
17        }
```

```
1  public double mapLinksandReward(VirtualNode vNode, SubstrateNode dstNode, HashMap<VirtualNode,SubstrateNode> map,
2  VirtualNetwork p,boolean finalMove) {
3              List<SubstrateLink> path;
4              double reward=0;
5              int conPath;
6              DijkstraShortestPath<SubstrateNode, SubstrateLink> shortestPath = new
7  DijkstraShortestPath<SubstrateNode, SubstrateLink>(
8                                    stack.getSubstrate()
9                        );
10             for(VirtualNode key: map.keySet()) {
11                   List<VirtualLink> links = new ArrayList<VirtualLink>();
12                   links.addAll(p.findEdgeSet(key, vNode));
13                   links.addAll(p.findEdgeSet(vNode,key));
14                   for(VirtualLink a: links) {
15                         VirtualNode dest=p.getDest(a);
16                         if(map.containsKey(dest)) {
17                               path=shortestPath.getPath(dstNode, map.get(dest));
18                         }
19                         else {
20                               path=shortestPath.getPath(map.get(key),dstNode);
21                         }
22                         conPath=0;
23                               for (SubstrateLink sl : path) {
24                                          for (AbstractDemand dem : a) {
25                                                for (AbstractResource res : sl) {
26
27                                                      if (res.accepts(dem) && res.fulfills(dem)) {
28                                                            conPath++;
29                                                                  dem.occupy(res);
30
31                                                      }
32                                                }
33
34                                          }
35                               }
36                               if(finalMove && path.size()==conPath && path.size()>0) {
37                                     mappedLinks++;
38                               }
39                               else if(!finalMove && path.size()==conPath && path.size()>0) {
```

```
40                                      processedLinks++;
41                                      if(path.size()>1)
42                                            reward=(2000/(path.size()))+reward;
43
44                                  }
45                                  else if (!finalMove && path.size()!=conPath && path.size()>0) {
46                                        reward=reward-2000;
47
48                                  }
49                        }
50                  }
51                  return reward;
52      }
```

```
1   private void getBestMapping(VirtualNetwork p, List<Integer>trainingMap) {
2               HashMap <VirtualNode, SubstrateNode> map = new HashMap <VirtualNode, SubstrateNode> ();
3               if(trainingMap!=null) {
4                     for(int i=0; i<states.size(); i++){
5                           if(nodeEmbedding(states.get(i),actions.get(trainingMap.get(i)))) {
6                           mapLinksandReward(states.get(i),actions.get(trainingMap.get(i)),map,p,true);
7                           map.put(states.get(i),actions.get(trainingMap.get(i)));
8                           }
9               }
10              }else {
11                    System.out.println(""+p.getName()+" NOT MAPPED");
12              }
13        }
```

- **Mètode getStates:** Genera una llista amb els estats de l'algorisme formats pels nodes de la xarxa virtual i els retorna, la llista canvia per cada xarxa virtual.

- **Mètode getActions:** Crea una llista amb les accions de l'algorisme formades pels nodes de la xarxa física.

- **Mètode fulfillDemand:** S'encarrega de comprovar si el node físic escollit pot assolir la demanda del node virtual determinat. Retorna false si no la pot assolir o true si la pot assolir.

- **Mètode epsilonCaculator:** Genera el valor d'epsilon corresponent a la iteració actual. Per això se li passa com a paràmetre el número d'iteracions, fa el càlcul corresponent a l'equació explicada en anteriors apartats i retorna epsilon.

- **Mètode Training:** És el mètode principal i entrena l'algorisme per elaborar una taula Q estable. Segueix el mateix mecanisme que l'explicat al pseudocodi: per cada iteració del while, fa un bucle amb un gran número d'iteracions, dintre un for itera sobre els diferents estats o nodes virtuals amb la corresponent decisió en cada estat entre exploració-explotació depenent del valor d'èpsilon i el node físic corresponent a l'acció escollida en aquell estat s'elimina de la llista d'accions disponibles, ja que no es permet allotjar més d'un node virtual sobre el mateix node físic. En l'ultima iteració del for s'afegeix una recompensa a cada parell estat-acció de la taula Q. Si l'allotjament ha sigut un èxit, la recompensa extra és positiva i sinó es negativa. Per últim, retorna una llista amb la millor acció corresponent a cada acció.

- **Mètode nodeEmbedding**: Comprova si el node virtual passat com a paràmetre pot ser allotjat en el node físic. Si es així, fa l'allotjament i retorna true i sinó retorna false, indicant que no s'ha fet l'allotjament.

- **Mètode mapLinksandreward**: Genera les recompenses per a l'acció escollida corresponent al seu node físic. Per això, com a paràmetre es passa el node físic en qüestió, el node virtual corresponent a l'estat actual, un mapa amb els allotjaments fets en anteriors estats (per trobar connexions entre els nodes ja allotjats i el node virtual per allotjar), la xarxa virtual sencera per poder obtenir els links (donat els nodes virtuals) i un booleà que diu si es tracta d'entrenament o de l'allotjament final. Primer es recorre el mapa dels allotjaments ja fets per veure si hi ha connexions amb el node virtual de l'acció escollida. En cas afirmatiu, per cada link virtual es troba el camí de la xarxa física més curt amb l'ús de l'algorisme de Dijkstra, donant com a paràmetre el node físic de l'acció escollida i el node físic on està allotjat el node virtual que manté la connexió amb l'actual. Si l'allotjament es pot fer, calcula les recompenses tenint en compte el número de salt i si no s'ha pogut allotjar, s'estableix una recompensa negativa. Pel contrari, si s'ha pogut allotjar en un sol link físic, s'incentiva amb una recompensa.

- **Mètode getBestMapping:** Rep una llista amb la millor acció o node físic escollit per cada estat o node virtual i la xarxa virtual actual, fa l'allotjament de cada node virtual en cada node físic segons la llista i es torna a utilitzar el mètode mapLinksandReward, però aquest cop com a paràmetre el booleà és true, ja que no és tracta d'entrenament.

4.2.3. Elecció els valors dels paràmetres

Escollir els valors dels paràmetres és una part important a l'hora d'executar un algorisme, en aquest cas el rati d'aprenentatge, que controla com de ràpid l'agent adopta els canvis aleatoris imposats per l'entorn, s'ha inicialitzat a 0.6, un valor entremig, ja que el sistema que s'esdevé del problema VNE es bastant complicat i es necessita de temps per poder fer proves per tal de no adoptar els canvis imposats per l'entorn massa ràpid. El valor mínim èpsilon, com es pot veure a la gràfica de la figura 10, és el valor mínim que prendrà la funció que calcula els valors d'èpsilon. El factor de descompte és un atenuant que fa decréixer la funció que calcula el valor d'èpsilon més lentament. El màxim d'iteracions és el valor màxim amb el qual l'agent podrà iterar sobre l'entorn en el que s'ha de fer l'allotjament.

Paràmetre	Valor
Rati d'aprenentatge (α)	0.6
Valor mínim Èpsilon (min_ ε)	0.01
Factor de descompte	0.001
Màxim iteracions (max_iter)	1000

Taula 3: Valors dels paràmetres

5. Resultats

Per a una correcta avaluació de l'algorisme implementat és necessària una bona comparació amb altres algorismes que també resolguin el problema. Per això s'han escollit tres algorismes més implementats en ALEVIN, que es detallaran a continuació.

5.1. Algorismes escollits per a comparar

- Detecció d'isomorfismes de subgrafs (SID) [18]: En aquest algorisme es proposa una solució basada en la teoria de grafs. Es tracta d'un algorisme heurístic basat en trobar un graf isomòrfic que representi una VNR que compleixi les demandes dins de la xarxa física. Una de les característiques d'aquesta proposta és la de fer l'allotjament de node i de links en el mateix escenari, el que ajuda a reduir temps d'execució quan es prenen decisions dolentes en l'allotjament de links. Aquest algorisme ha estat escollit per comparar-se amb l'algorisme implementat en termes de temps d'execució, ja que els

seus desenvolupadors asseguren que el mapeig amb és molt ràpid degut a que tot es fa sobre el mateix escenari. També asseguren que és molt adequat pel mapeig de xarxes virtuals grans, perfectes per avaluar el temps d'execució dels algorismes.

- VN bàsic [19]: En aquest algorisme es vol optimitzar l'estrès de link i node de forma independent dividint-ho en dos problemes més petits per resoldre-ho de forma seqüencial. Per tant, es proposa una assignació de xarxes virtuals que consideri l'estrès de nodes i links a tot el procés d'assignació de la xarxa virtual. Aquest algorisme ha estat escollit perquè els seus desenvolupadors asseguren que té un molt bon rendiment en xarxes amb un gran número de condicions, també asseguren un molt bon rendiment en xarxes que no estan gaire connectades com és el cas de les que estan formades de manera aleatòria, sumat a que divideix el problema en dos més petits per optimitzar el nivell d'estrès serà una bona comparació en termes de temps d'execució.

- BFS node ranking Coordinated mapping [20]: Aquest algorisme consta només d'un escenari de mapeig, fa servir l'estratègia de backtracking i l'algorisme de cerca en amplada per recórrer els elements del graf, per allotjar els nodes i links virtuals en el mateix escenari. La idea de emprar aquest algorisme és l'estalvi en ample de banda dels links de la xarxa física al no haver d'utilitzar dos escenaris per fer el mapeig. Aquest algorisme ha estat escollit per comparar-se amb l'algorisme implementat en termes d'acceptació de xarxes virtuals i de Revenue a llarg termini, ja que els seus desenvolupadors asseguren un rendiment molt alt de l'algorisme en aquestes dues mètriques a costa de temps d'execució degut al nivell alt de computació.

5.2. Topologies de Xarxa escollides per a les simulacions

Les simulacions fetes per avaluar l'algorisme implementat s'han dut a terme sobre tres escenaris diferents, els quals utilitzen dos tipus de topologies per construir les seves xarxes, la topologia de quadricula i la aleatòria:

- Topologia de quadricula: Aquesta topologia fa que les xarxes tinguin forma de quadricula. Una bona opció per obtenir estructures de xarxes que s'empren en casos reals on no es requereix d'una gran complexitat de xarxa perquè els costos han de ser baixos i obtenen un bon rendiment.

- Topologia aleatòria: Aquesta topologia crea xarxes aleatòries. El generador que s'encarrega de fer aquestes topologies és Waxman [21]. La seva tasca és modelar topologies de xarxa amb el propòsit d'avaluar algorismes d'encaminament. En els grafs de Waxman els nodes estan uniformement distribuïts sobre una àrea rectangular i els links són afegits entre els nodes a partir d'un mecanisme aleatori. Un dels usos més importants per aquesta topologia feta amb el generador Waxman és el modelat de les xarxes intra-domini, les quals s'utilitzen a les simulacions per avaluar el rendiment dels algorismes. A aquest generador se li han de proporcionar dos

paràmetres: α i β. El primer incrementa la probabilitat de generar links entre dos nodes i el segon incrementa la probabilitat d'obtenir links més grans.

Amb aquests tipus de topologies es pretén simular escenaris reals on es fan servir aquests tipus de xarxes i avaluar les mètriques més importants a cada un d'ells.

5.3. Mètriques escollides per a la avaluació del algorisme

Per avaluar l'algorisme i comparar els resultats obtinguts amb els altres algorismes s'han escollit les mètriques més importants emprades a l'avaluació del rendiment dels algorismes que ressolen el problema de VNE:

- Cost/Revenue: Aquesta mètrica es farà servir per avaluar com optimitza l'algorisme els recursos de la xarxa física per trobar allotjaments. Un valor alt d'aquesta mètrica significa que es necessiten molts recursos per a fer l'allotjament de les xarxes virtuals. L'allotjament perfecte seria el que obtingués una relació Cost/Revenue de 1.
    - Cost: Fa referència al total de recursos físics utilitzats per fer el mapeig de les xarxes virtuals. Es determina tenint en compte tots els recursos de la xarxa física que s'han reservat per les VNRs.
    - Revenue: Fa referència a la suma dels recursos virtuals sol·licitats per les xarxes virtuals a les que s'ha fet el mapeig.
- Temps d'execució o complexitat: Aquesta mètrica es farà servir per avaluar el temps que triga l'algorisme en dur a terme les simulacions.
- Acceptació de VNRs: Aquesta mètrica s'emprarà per avaluar la capacitat de l'algorisme en allotjar diferents VNs sobre una mateixa xarxa física, tenint en compte l'aprofitament dels recursos de la xarxa.

5.4. Escenaris de simulació

Per obtenir les millors mètriques, s'han escollits diferents escenaris pel correcte estudi de cadascuna d'elles. A continuació es detallaran els principals aspectes de cada escenari i quina és la mètrica estudiada a cada un d'ells sobre els diferents algorismes.

5.4.1. Primer escenari

Aquest escenari consta d'una xarxa física amb nou nodes formant una topologia de quadricula. S'escull aquesta topologia per tal d'obtenir un tipus de xarxes semblants a les xarxes mallades que tant es fan servir per millorar les prestacions d'una xarxa i el que es pretén en aquest escenari és, a partir de la mateixa xarxa física, fer diferents simulacions

incrementant la mida de la xarxa virtual a allotjar per tal d'avaluar la mètrica de cost/revenue amb el supòsit d'obtenir un bon rendiment en termes de reduir costos a l'allotjament, ja que tots els nodes de la xarxa estan interconnectats entre si.

En aquest escenari els nodes i links de la xarxa física tenen 100 de recursos en CPU i ample de banda, respectivament. La VN de cada subescenari incrementa el número de nodes en dos a cada subescenari, les demandes de la VN van de 0 a 90 per cada link i node en termes d'ample de banda i CPU, respectivament.

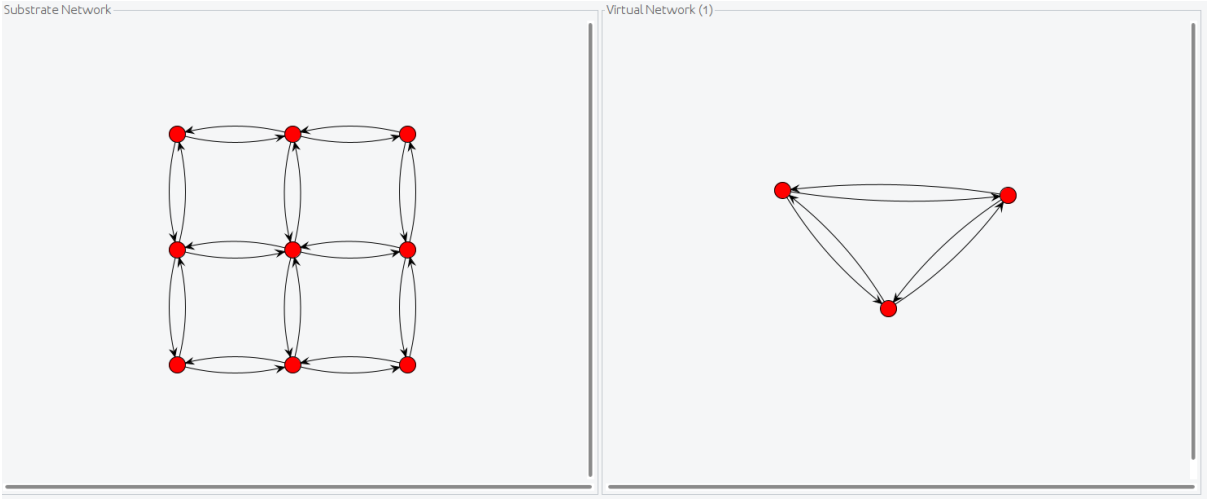

Figura 13: Escenari 1.1 simulació

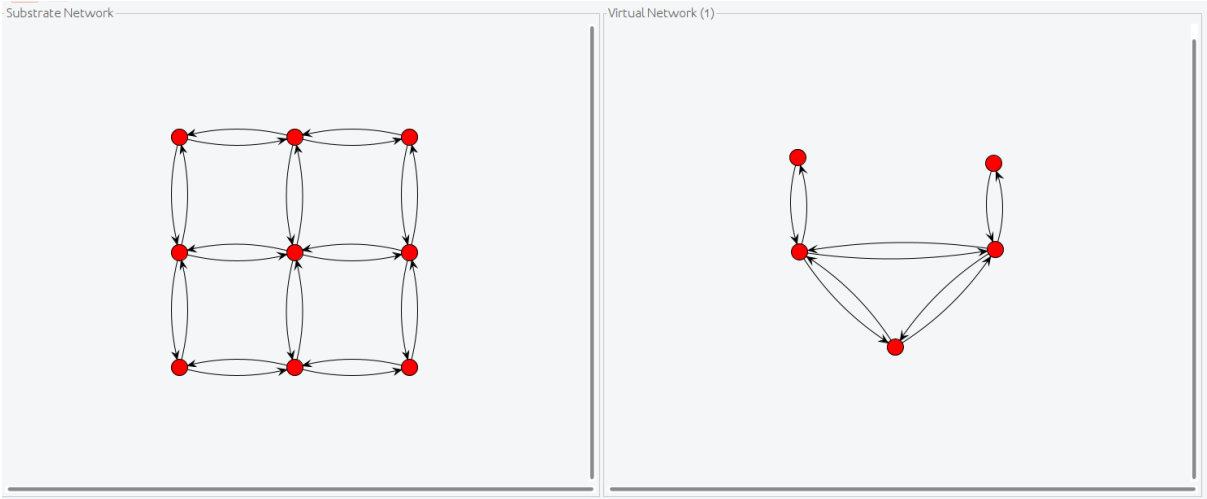

Figura 14: Escenari 1.2 simulació

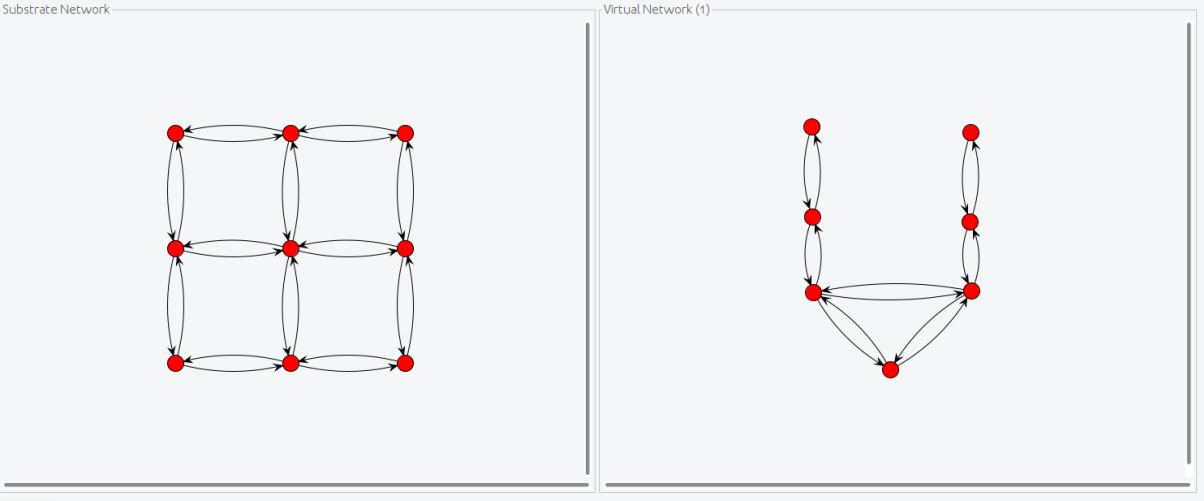

Figura 15: Escenari 1.3 simulació

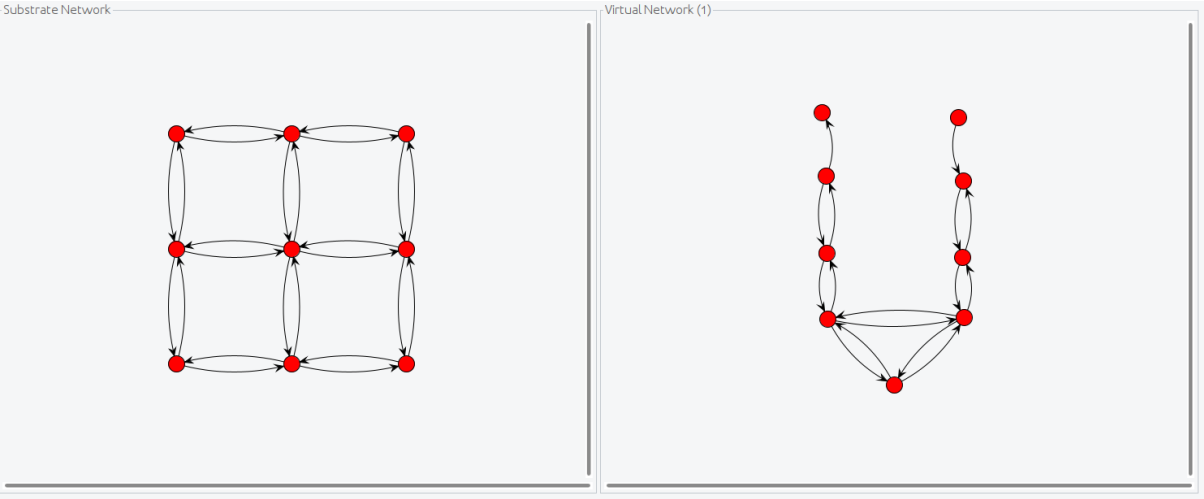

Figura 16: Escenari 1.4 simulació

COST/REVENUE	Q LEARNING	BFS NODE RANKING	BASIC VN	SUBGRAPH ISO
escenari 1.1(figura 13)	1.222	1.844	1.800	1.800
escenari 1.2 (figura 14)	1.123	1.346	1.590	1.716
escenari 1.3 (figura 15)	1,085	1.085	1.468	1.632
escenari 1.4 (figura 16)	1,109	1.126	1.414	No allotjat

Taula 4: Cost/Revenue escenaris 1.x

A la taula 4 es mostren els resultats que fan referència al cost/revenue dels subescenaris als diferents algorismes. En anteriors apartats ja s'ha esclarit que la mètrica cost/revenue fa referència a la divisió entre els recursos de la xarxa física necessaris per a l'allotjament i els recursos demandats per la xarxa virtual. El valor òptim a obtenir hauria de ser baix, traduït com a un cost baix i un revenue alt, el que fa que el valor de la divisió sigui petit.

A la gràfica de la figura 17 es pot veure que tots els algorismes tendeixen a la baixa a mesura que l'allotjament de la VNR és més gran. Per a la primera dada de la gràfica, que és l'execució de l'allotjament amb una xarxa virtual de 3 nodes (figura 13), s'aprecia una clara diferència entre l'algorisme pel que s'aposta en aquest treball i els altres 3 escollits a la comparació que tenen un valor més semblant. Com a cloenda podem dir que la primera resposta que s'obté és que l'algorisme implementat supera als altres en l'allotjament d'una xarxa virtual petita en termes d'aprofitament dels recursos.

Per a la segona dada de la gràfica, l'allotjament d'una VN amb 5 nodes (figura 14), es veu com tots els algorismes obtenen un valor més petit al primer, destacant l'algorisme BFS node ranking coordinated mapping que obté un molt bon resultat, tot i que l'algorisme implementat segueix obtenint millors resultats.

Amb la tercera dada de la gràfica, l'allotjament d'una VN amb 7 nodes (figura 15), segueix la tendència a la disminució dels valors de la mètrica en tots els algorismes, en el cas de l'algorisme BFS node ranking coordinated mapping obté un resultat igual al de l'algorisme implementat. Els altres dos algorismes encara segueixen tenint pitjors valors en comparació amb els primers.

En referència a l'última dada de la gràfica, l'allotjament d'una VN amb 9 nodes (figura 16), cal destacar que aquesta VN té els mateixos nodes que la xarxa física, el que fa que algun algorisme com SID no sigui capaç de dur a terme l'allotjament. En aquest allotjament es pot veure com els algorismes amb millors resultats fins ara Q learning i BFS empitjoren lleugerament. En cas del BFS, empitjora una mica més i el VN basic millora respecte a l'anterior dada per molt poc.

En termes d'estabilitat, s'ha de valorar que l'algorisme implementat va obtenint millors valors en cada allotjament, destacant el primer valor obtingut quan s'han executat els algorismes per allotjar una xarxa de tres nodes (figura 13). Això vol dir que aquest algorisme té molta cura de la forma que s'utilitzen els recursos de la xarxa física i es podria dir que un dels seus punts forts és l'eficiència amb la que es tracten els recursos de la xarxa en cada allotjament, obtenint les solucions més òptimes possibles.

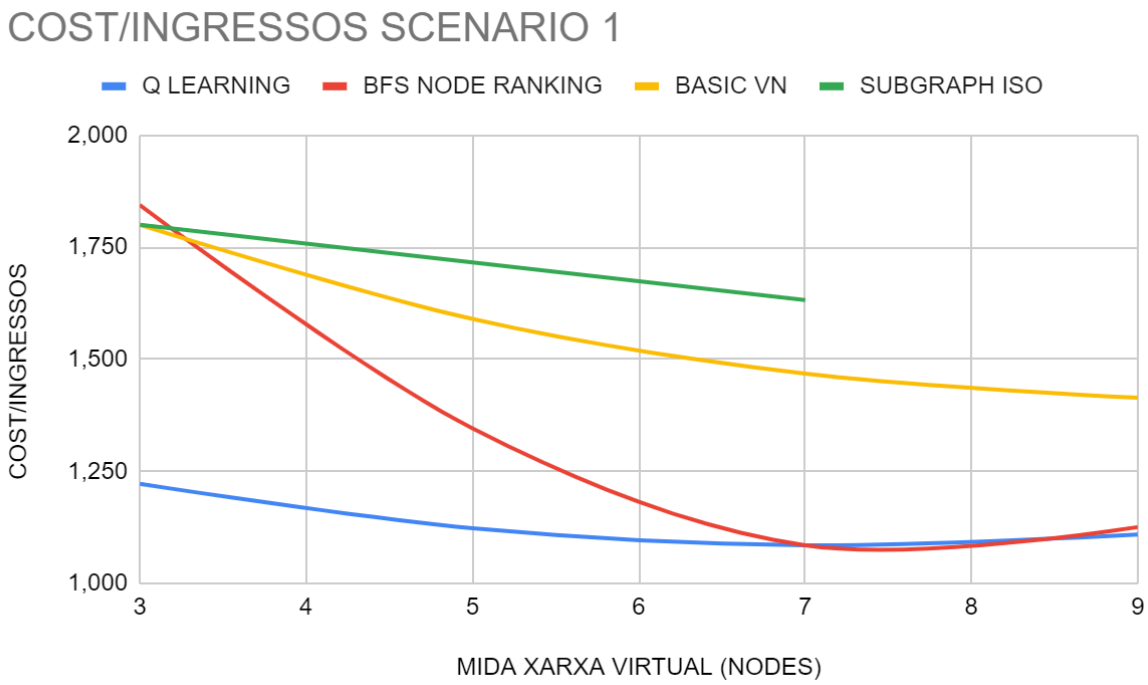

Figura 17: Gràfica cost/Revenue escenaris 1.x

5.4.2. Segon escenari

Aquest escenari consta d'una xarxa física amb deu nodes formant una topologia amb forma aleatòria amb el generador Waxman (α=1 , β=0.5). El que es vol aconseguir amb aquest escenari és avaluar la mètrica d'acceptació de xarxes virtuals, per això es volen construir diferents subescenaris amb la mateixa xarxa física i en cada un d'ells incrementar el número de xarxes virtuals que s'han d'allotjar en un, començant amb el primer subescenari amb una VNR, el segon amb dos VNR fins al sisè amb sis VNR, amb l'objectiu de veure la quantitat de VNR acceptades en cada algorisme i comparar-ho amb la mètrica obtinguda amb Q learning.

Emprar el generador Waxman en aquests escenari és clau per obtenir una bona avaluació de l'algorisme respecte a la mètrica d'acceptació de xarxes virtuals, ja que fer l'allotjament de diferents xarxes aleatòries sobre una aleatòria és tot un repte per als algorismes utilitzats.

La xarxa física consta de 250 recursos màxim en cada node de CPU i 250 recursos màxim d'ample de banda en cada link, el que vol dir que cada node i link físic respectivament obtindrà un número de recursos entre 0 i 250 de forma aleatòria. Cada VN conté 3 nodes i de 3 a 5 links cada una, 70 de recursos com a màxim en CPU per a cada node i 70 de recursos com a màxim en ample de banda per a cada link, obtenint un número de recursos entre 0 i 70 de forma aleatòria cada element de la xarxa virtual.

D'aquesta manera, es pot avaluar la capacitat dels algorismes per allotjar diferents xarxes sobre la mateixa xarxa física on els recursos es distribueixen de forma aleatòria, posant a proba el rendiment dels mateixos.

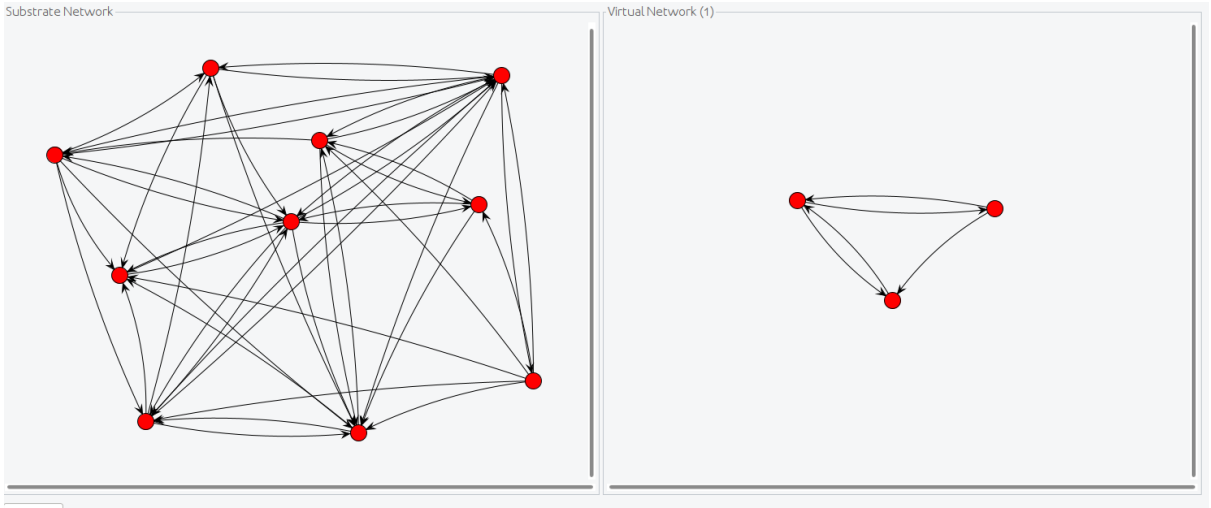

Figura 18: Escenari 2.1 simulació

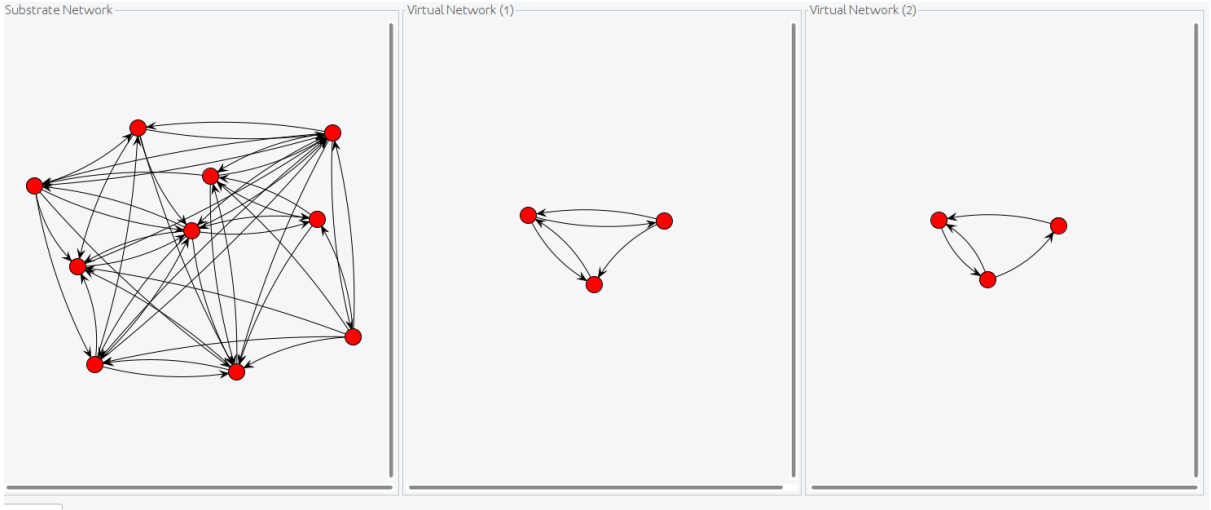

Figura 19: Escenari 2.2 simulació

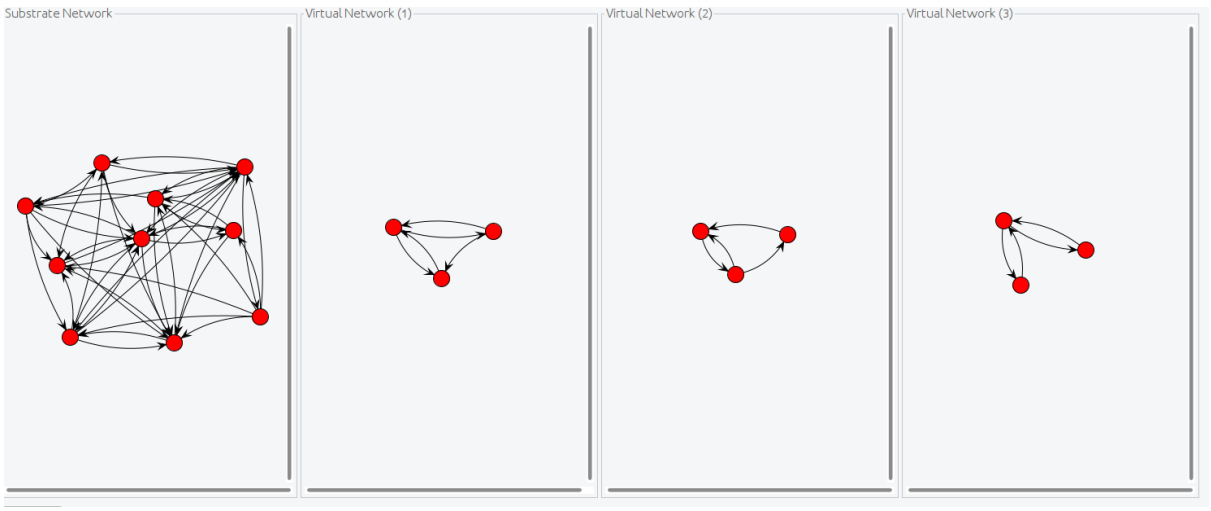

Figura 20: Escenari 2.3 simulació

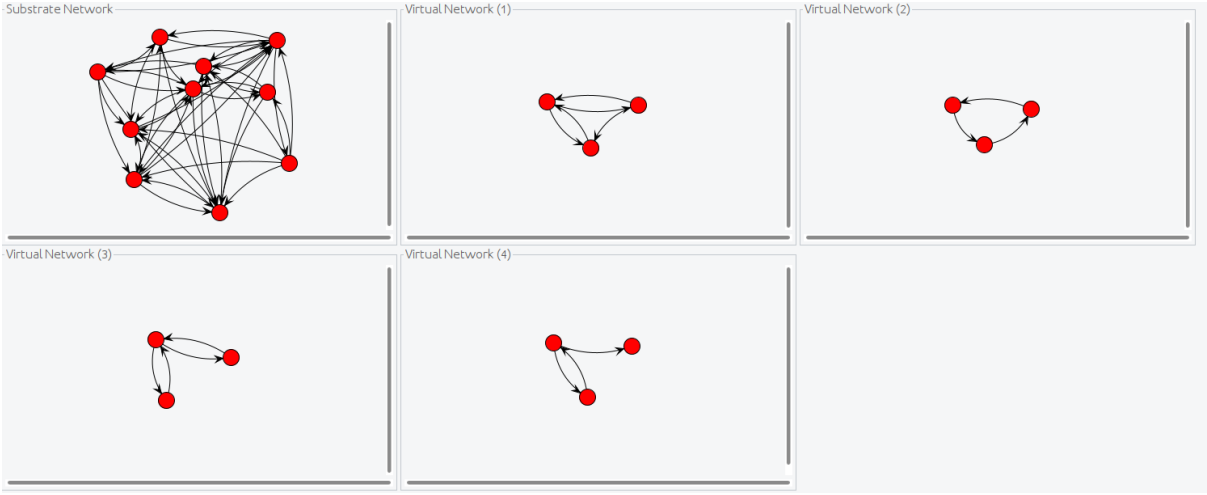

Figura 21:Escenari 2.4 simulació

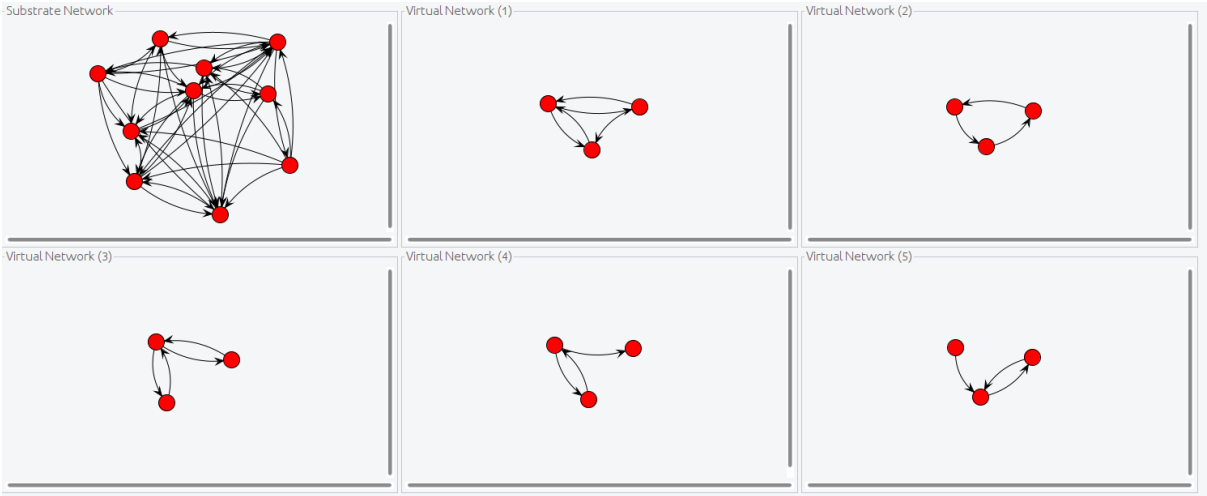

Figura 22: Escenari 2.5 simulació

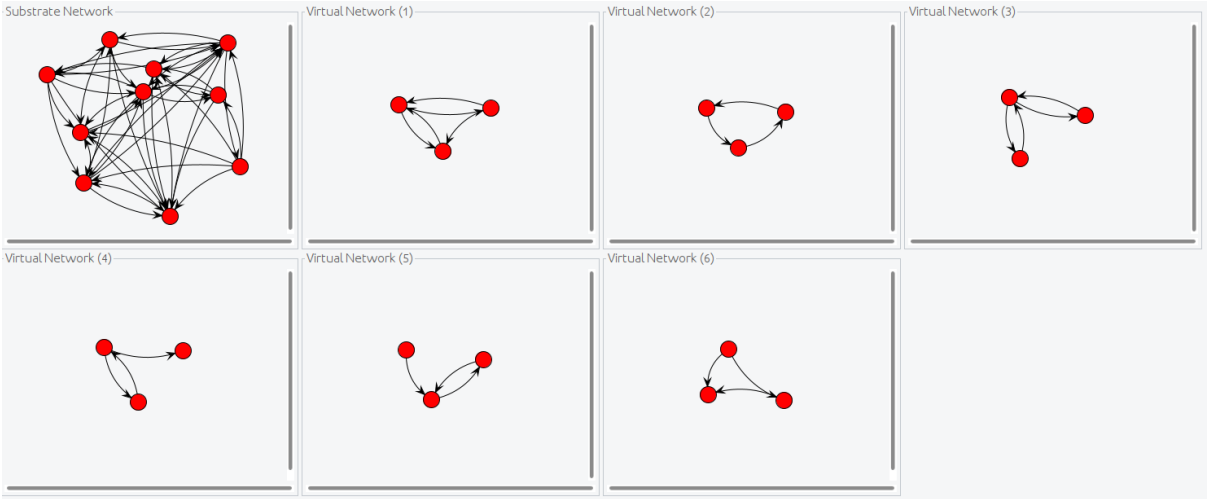

Figura 23: Escenari 2.6 simulació

A la taula 5 es poden observar els resultats obtinguts en cada execució per cada algorisme.

A la primera execució on s'allotja una VNR com es pot veure a la Figura 18, cap dels algorismes té problemes en fer l'allotjament de la VNR.

A la segona execució es pretén allotjar dos VNRs (Figura 19). En aquest cas es pot observar com l'algorisme BFS Node ranking ja té problemes amb l'allotjament de la segona VNR, en canvi els altres algorismes segueixen allotjant totes les VNR.

A la tercera execució es busca allotjar tres VNRs (Figura 20). En aquesta execució Q learning segueix sent capaç d'allotjar totes les VNR, BFS node ranking torna a poder allotjar les tres VNRs, en canvi Basic VN i SID només poden allotjar la primera VNR de les tres possibles.

A la quarta execució es vol allotjar quatre VNRs (Figura 21). En aquest cas, cap dels algorismes presenta problemes per allotjar les quatre VNRs.

A la cinquena execució es pretén allotjar cinc VNRs (Figura 22). En aquesta ocasió, estem parlant ja d'allotjar 15 nodes sobre els 10 de la xarxa física, on Q learning segueix sent capaç d'allotjar la totalitat de VNRs, Basic VN només en pot allotjar dos, subgraph iso allotja quatre de les cinc i BFS node ranking torna a poder allotjar totes.

A l'última execució es vol allotjar sis VNRs (Figura 23), el que vol dir 18 nodes en els 10 de la xarxa física. En aquest cas, l'únic algorisme capaç d'allotjar totes les demandes es Qlearning, BFS node ranking allotja cinc de sis VNR, Basic VN allotja 2 de sis VNR i SUbgraph ISO allotja cinc de sis VNR.

ACCEPTACIÓ VNRs	Q LEARNING	BFS NODE RANKING	BASIC VN	SUBGRAPH ISO
escenari 2.1 (figura 18)	1	1	1	1
escenari 2.2 (figura 19)	1	0.50	1	1
escenari 2.3 (figura 20)	1	1	0.33	0.33
escenari 2.4 (figura 21)	1	1	1	1
escenari 2.5 (figura 22)	1	1	0.4	0.8
escenari 2.6 (figura 23)	1	0.83	0.33	0.83

Taula 5: Acceptació de VNRs escenaris 2.x

La gràfica de la figura 24 recopila les dades de la taula 5, explicada punt per punt. A la gràfica es pot veuré com l'algorisme implementat, Q learning, es manté constant a cada execució allotjant totes les VNR, en canvi els altres algorismes no tenen aquesta constància, ja que no són capaços d'allotjar la totalitat de les VNRs en alguna de les execucions. Per tant, Q learning, en aquest cas, és el millor algorisme dels quatre en termes d'allotjament de diferents VNRs.

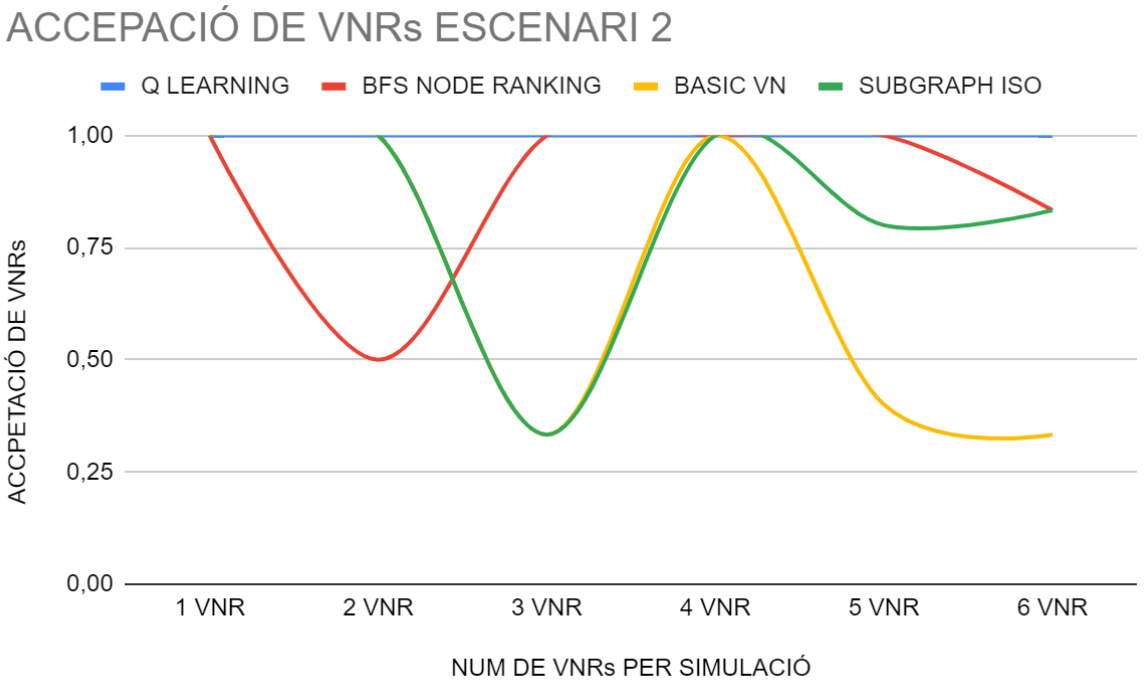

Figura 24: Gràfica acceptació de VNRs escenaris 2.x

5.4.3. Tercer escenari

Aquest escenari consta d'una xarxa física amb 30 nodes formant una topologia de xarxa aleatòria amb el generador Waxman (α=1 , β=0.5). El que es pretén és, a partir de la mateixa xarxa física, fer diferents simulacions incrementant la mida de la xarxa virtual a allotjar, per tal d'avaluar la mètrica de temps d'execució. Com el problema de VNE pertany als de tipus NP-difícil i els temps d'execució augmenten de forma exponencial amb la mida de les xarxes, s'avaluarà el rendiment de l'algorisme en termes de complexitat amb xarxes mitjanes emprant les topologies aleatòries que tant s'utilitzen en xarxes intra-domini per a l'avaluació de les mateixes, esdevenint un repte per als diferents algorismes allotjar una xarxa aleatòria en una altre de mida molt gran.

Cada subescenari estarà format només per una VNR i la xarxa física en qüestió, de tal manera que en cada subescenari s'anirà incrementant el número de nodes començant el primer subescenari amb 3 nodes fins l'últim amb 10 nodes.

Els recursos de la xarxa física es reparteixen de forma aleatòria en cada node i link amb un valor entre 0 i un màxim de 300 recursos. Els de la VNR es reparteixen en cada node i link amb un valor entre 0 i un màxim de 50 recursos. En aquest escenari no es vol obtenir cap impediment en l'allotjament per culpa dels recursos demandats.

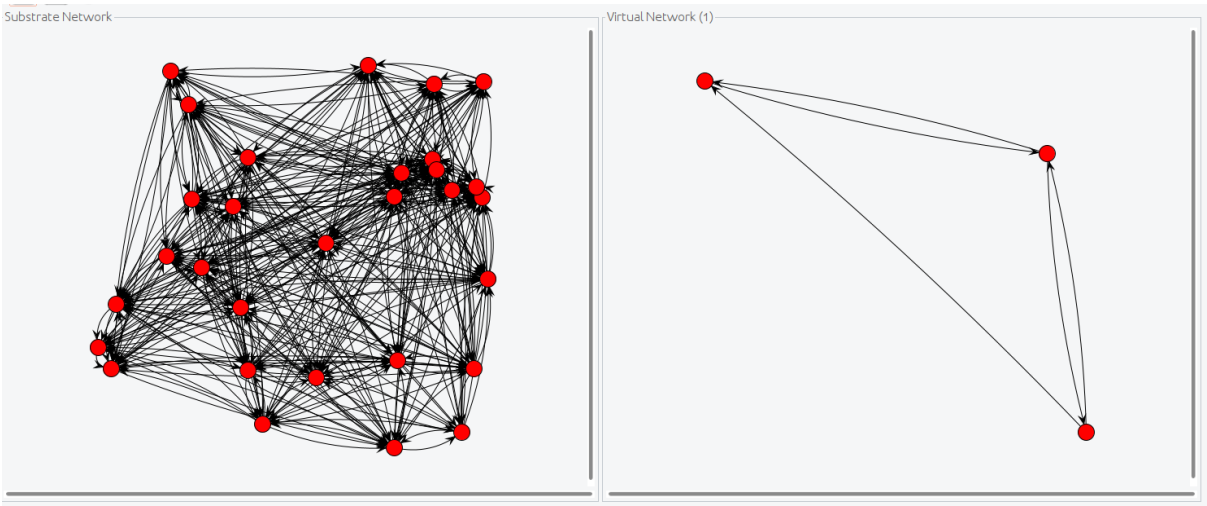

Figura 25: Escenari 3.1 simulació

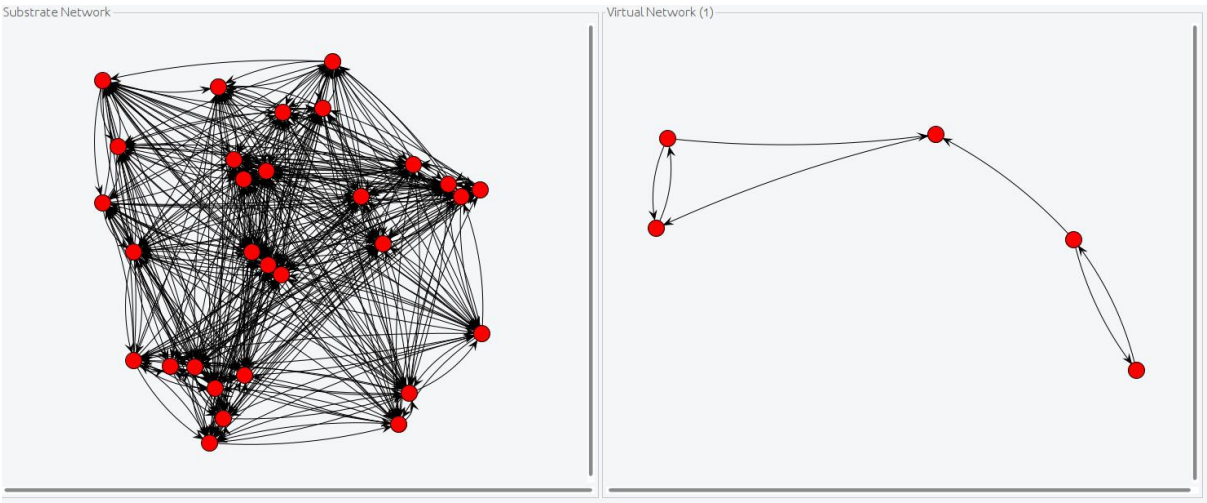

Figura 26: Escenari 3.2 simulació

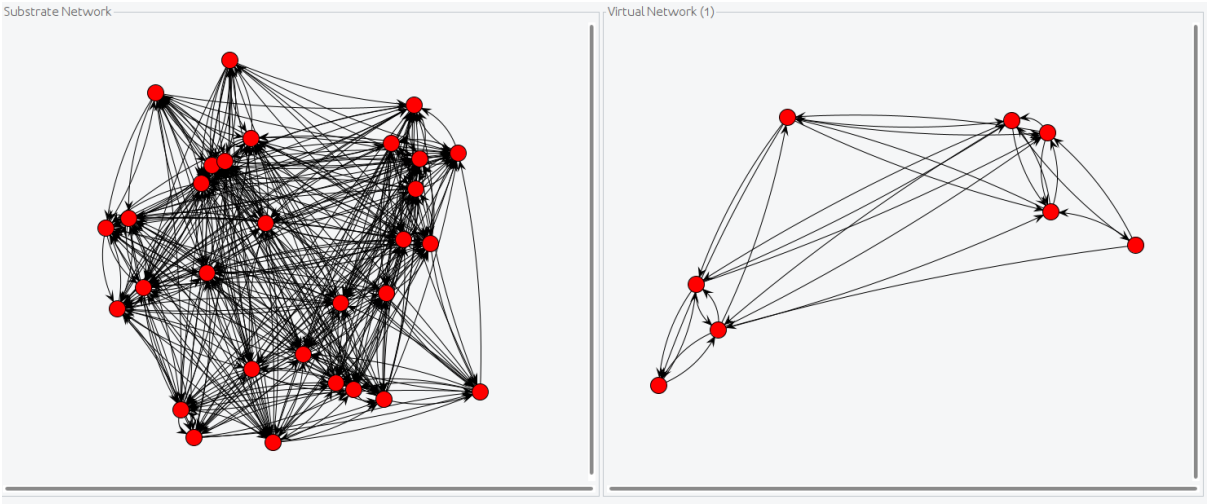

Figura 27: Escenari 3.3 simulació

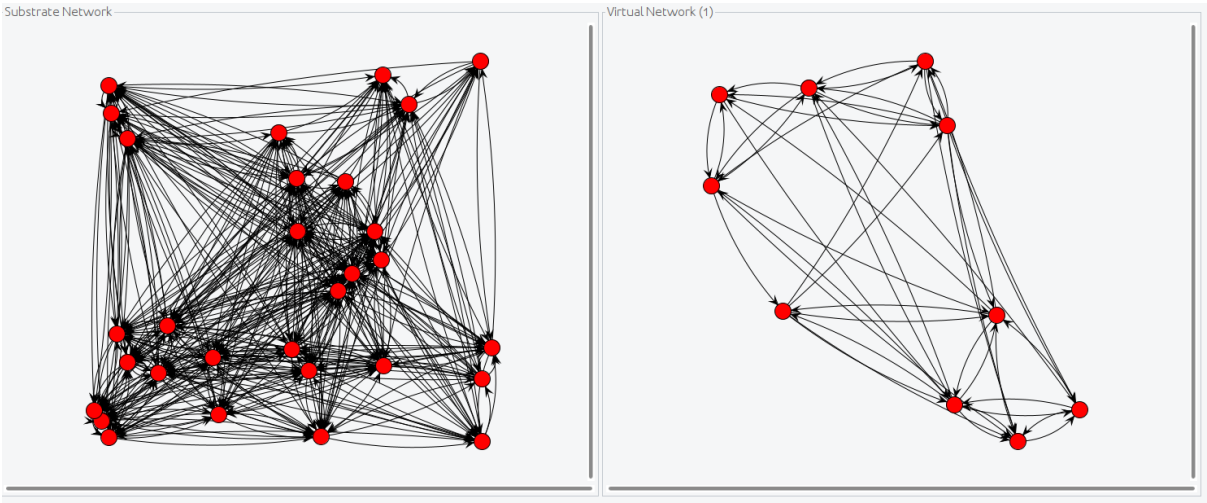

Figura 28: Escenari 3.4 simulació

A la taula 6 es poden observar els resultats obtinguts en cada execució per cada algorisme.

A la primera execució on s'allotja la VNR amb 3 nodes, com es pot veure a la Figura 25**,** cap dels algorismes té problemes en fer l'allotjament de la VNR, però l'algorisme Q learning i el de BFS node ranking coordinated VNE obtenen valors de l'ordre de 25 o 30 cops superiors respecte als altres dos.

A la segona execució es pretén allotjar la VNR amb 5 nodes (Figura 26) i en aquest cas es pot observar com els quatre algorismes obtenen valors superiors als anteriors.

A la tercera execució es busca allotjar la VNR amb 8 nodes (Figura 27). Aquí es pot veure com el temps d'execució en tots els algorismes es torna a multiplicar.

A la quarta execució s'allotja la VNR amb 10 nodes (Figura 28). En aquesta execució tots els algorismes continuen obtenint temps d'execució superiors, excepte l'algorisme BFS node ranking coordinated VNE que aconsegueix estabilitzar el temps aconseguit en l'anterior execució.

TEMPS EXECUCIÓ	Q LEARNING	BFS NODE RANKING	BASIC VN	SUBGRAPH ISO
escenari 3.1 (figura 26)	0.7	0.53	0.018	0.008
escenari 3.2 (figura 27)	0.95	0.84	0.051	0.012
escenari 3.3 (figura 28)	2	1.2	0.15	0.029
escenari 3.4 (figura 29)	3.28	1.067	0.28	0.072

Taula 6: Temps d'execució escenaris 3.x

A la taula 6 es poden apreciar els temps d'execució dels diferents algorismes i tal com es pot apreciar a la gràfica de la figura 29 basic VN i SID obtenen temps d'execucions molt bons i una projecció que en escenaris grans no s'incrementarà aquest de forma tan abrupte. D'altra banda, BFS node ranking coordinated VNE i Q learning obtenen temps pitjors, esdevenint Q learning lleugerament pitjor a BFS node ranking, ja que com s'ha vist l'algorisme fa moltes iteracions per obtenir els millors allotjament.

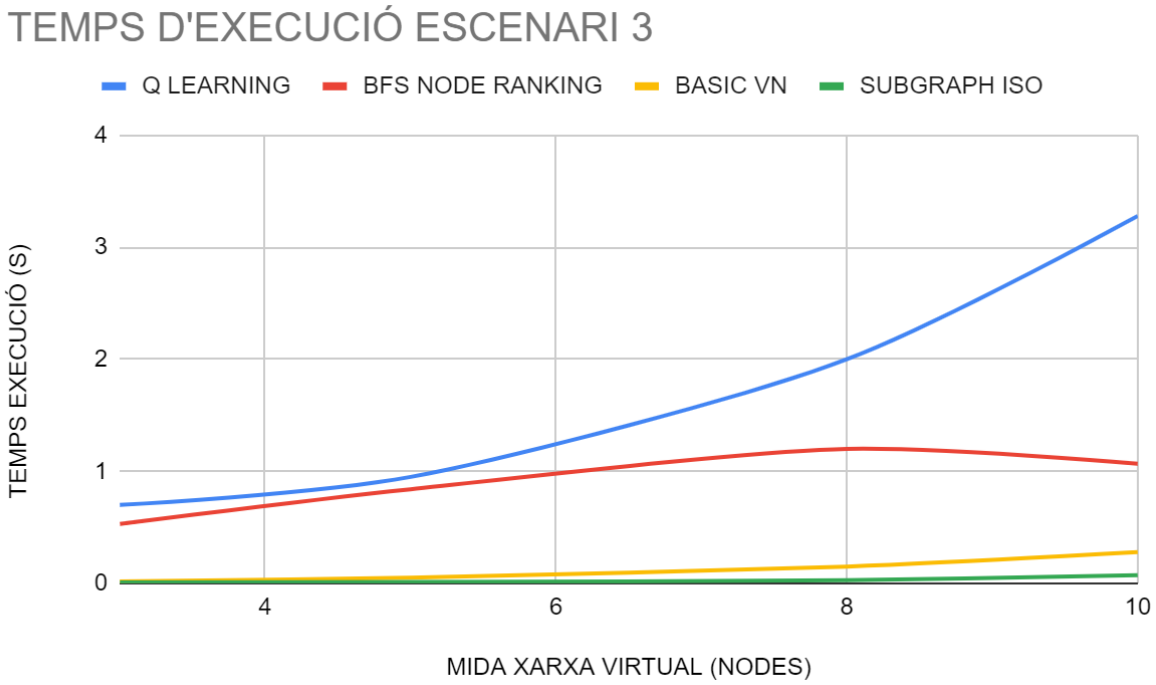

Figura 29: Gràfica temps d'execució escenaris 3.x

6. Pressupost

En aquest apartat es mostrarà un petit anàlisis dels costos del projecte. Per fer el càlcul del pressupost del projecte s'ha de tenir tres aspectes en compte.

- Material: En la següent taula es fa un breu resum dels materials fets servir al projecte. Pels materials de tipus immoble s'utilitza una vida útil de vuit anys, mentre que pels de tipus tecnològic s'empra un vida útil aproximada d'uns cinc anys per a calcular la depreciació de cada element.

Material	**Cost**	**Depreciació**
Cadira oficina	200€	8.33€
Taula d'oficina	250€	10.42€
Ordinador	600€	40€
Perifèrics de l'ordinador	100€	6.67€
Total	**1150€**	**65.42€€**

Taula 7: Estimació dels costos materials

- Mà d'obra: Per calcular els costos en mà d'obra del projecte es fa servir com a model un estudiant de telecomunicacions amb un salari de 12€/h. El treball repercuteix en un total d'unes 450 hores, equivalents als 18 ECTS dels que consta el projecte, i esdevenint en un cost final equivalent a **5400€**.

- Costs indirectes: Els costs indirectes es calcularan a partir de l'ús d'internet i electricitat al projecte, fent una mitjana d'uns 35€ per mes en internet i una mitjana d'uns 40€ per mes en electricitat. En conclusió, el cost indirecte és de **300€**.

A fi de poder mostrar les dades d'una forma molt més clara, es resumeixen el total dels costos per a desenvolupar aquest projecte a la taula següent:

Concepte	Cost
Material	65.42€
Ma d'obra	5400€
Costs indirectes	300€
Cost total	**5765.42€**

Taula 8: Estimació dels costos totals

7. Conclusions

En aquesta secció s'avaluarà la solució trobada al problema VNE emprant les eines proveïdes per la IA, també es veuran els marges de millora de l'algorisme implementat.

7.1. Utilitat IA en VNE

Al llarg d'aquesta memòria s'han avaluat les tècniques existents d'aprenentatge en la intel·ligència artificial per trobar-ne la tècnica més apta per a solucionar el problema de VNE, a partir de la implementació feta amb l'algorisme Q learning i comparant-lo amb altres algorismes. La IA es presenta com un mitjà pràctic per a solucionar el problema VNE, aportant tècniques innovadores com la capacitat d'entrenar-se, aprendre i resoldre.

Escollir algorismes pertanyents a la IA comporta una sèrie de peculiaritats en la seva implementació respecte altres algorismes. Per exemple, una peculiaritat observada, i que tots els algorismes de IA comparteixen, és la de seguir un patró d'entrenament, sigui el tipus d'aprenentatge que sigui amb dades etiquetades, sense etiquetar o les generades pel propi entorn. Degut a aquest patró d'entrenament, l'algorisme obté una capacitat de reacció, memòria i aprenentatge. Una altra peculiaritat d'escollir algorismes pertanyents a la IA és la seva gran adaptabilitat en les variacions que poden haver en un problema, o el seu gran rendiment permetent el maneig de grans quantitats d'informació

7.2. Avaluació dels resultats

L'algorisme implementat ha estat sotmès a la simulació de diferents escenaris per tal d'avaluar el seu rendiment amb diverses mètriques. D'aquesta avaluació es poden obtenir els següents avantatges i desavantatges:

- Avantatges
    - Els costos són baixos respecte altres algorismes tradicionals.
    - Revenue baix respecte altres algorismes tradicionals.
    - Els recursos de la xarxa física s'utilitzen de la forma més òptima possible.
    - Alta capacitat d'acceptació de VNRs respecte altres algorismes tradicionals.
- Desavantatges
    - Els temps d'execució són superiors respecte altres algorismes tradicionals.

- L'execució de simulacions amb xarxes grans pot derivar en temps d'execucions molt grans.

A l'elaboració dels resultats s'han fet servir tres algorismes més, ja implementats a l'ALEVIN, del que sorprèn el baix temps d'execució dels algorisme SID i basic VN, tot i que aquests no presenten una certa estabilitat en termes d'acceptació de xarxes virtuals i tenen una relació de cost/revenue pobre respecte dels altres dos algorismes.

L'algorisme BFS node ranking coordinated mapping obté temps d'execució lleugerament inferiors a l'algorisme implementat, però superiors als de SID i basic VN. D'altra banda, obté millors resultats en les altres dues mètriques respecte a SID i basic VN, però pitjors resultats que l'algorisme implementats, el que no el fa destacar en cap aspecte.

La modelització del problema de VNE amb un algorisme provinent de la IA obre noves vies d'estudi de l'eficiència dels algorismes tradicionals que s'utilitzen a la resolució del problema, ja que aquests presenten unes mètriques pitjors en termes de cost/revenue i acceptació de xarxes virtuals.

Els bons resultats obtinguts per Q learning no són casualitat, al llarg d'aquest projecte s'ha mostrat com aquest algorisme està enfocat a la millora d'aquestes dues mètriques, gràcies al mètode d'entrenament implementat amb les corresponents recompenses obtingudes amb l'allotjament de links virtuals i amb els incentius donats al final de cada iteració depenent de si l'allotjament ha sigut un èxit o no. S'ha de tenir en compte, no obstant, el gran número d'iteracions que necessita l'algorisme per trobar la resolució més òptima al problema, però també que ha tingut les millors mètriques de la comparativa en cost/revenue i en acceptació de xarxes virtuals.

7.3. Treball futur

L'algorisme implementat es mostra com un dels millors mitjans per a solucionar l'allotjament de xarxes virtuals petites i mitjanes en termes de acceptació de xarxes i cost/revenue, però degut a la utilització de taules per guardar els valors Q calculats mitjançant la parella estat-acció i el gran número d'iteracions que empra l'algorisme a l'entrenament fa que en termes de temps d'execució s'obtinguin resultats pobres, els pitjors de la comparativa. A més, Q learning no està pensat per a resoldre problemes en entorns molt grans, ja que s'obtindria una taula amb milers d'estats i accions, que derivaria en temps d'execució no assumibles.

Trobar marges de millora en un algorisme en termes de temps d'execució es presenta com un repte difícil, però apassionant. Si es vol obtenir un millor rendiment en termes de temps d'execució, podrien explorar-se dues vies:

- Cercar un nou algorisme de IA que resolgui el problema de VNE, el que significa que tot el progrés fet en termes de cost/revenue i acceptació de xarxes virtuals podria no ser el mateix.

- Millorar l'algorisme amb noves eines per poder obtenir millors resultats en termes de temps d'execució, però mantenint els bons resultats obtinguts a les altres mètriques, amb l'objectiu d'obtenir un millor rendiment en termes de temps d'execució de l'algorisme.

Com a marge de millora, l'opció més òptima és explorar aquestes eines de IA que es presenten com una actualització a l'algorisme implementat:

Deep-Q learning[22] és una gran oportunitat per a millorar l'algorisme implementat. Aquest algorisme és una extensió de l'algorisme clàssic de Q learning, fa servir les xarxes neuronals profundes per aproximar els valors de Q, on els estats formen l'input i el valor Q de totes les accions possibles es genera com l'output. A la figura 30 es fa el mapeig d'una xarxa neuronal on l'input són els estats amb valor [1,2,3,4] i la sortida es representa amb les accions amb el seu corresponent valor Q. Fer servir xarxes neuronals en detriment de les taules Q per guardar el valor Q de cada parella estat-acció fa que el temps de computació necessari per executar l'algorisme sigui menor.

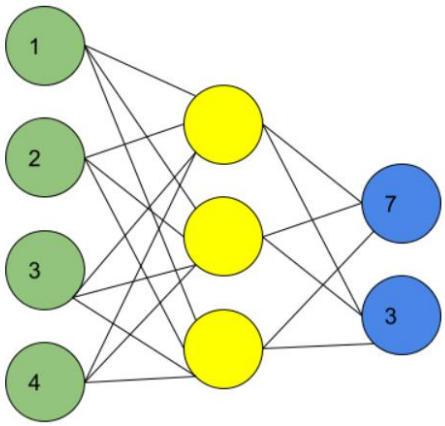

Figura 30: Mapeig d'una xarxa neuronal

Incloure aquesta extensió a l'algorisme clàssic Q learning en un futur milloraria les prestacions en termes de temps d'execució i mantindria els bons resultats en termes de cost/revenue i acceptació de xarxes virtuals, ja que el cos de l'algorisme continuaria sent el mateix perquè només canviaria la manera de guardar la informació.

Bibliografia

[1] Stephen Watts, “What is a Virtual Network?,” *https://www.bmc.com/blogs/virtual-network/#*, Jun. 16, 2020. https://www.bmc.com/blogs/virtual-network/# (accessed Sep. 22, 2021).

[2] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, “Network virtualization: Technologies, perspectives, and frontiers,” *Journal of Lightwave Technology*, vol. 31, no. 4. pp. 523–537, 2013. doi: 10.1109/JLT.2012.2213796.

[3] B. Andriy Burkov, *The Hundred-Page Machine Learning*. 2019. Accessed: Sep. 23, 2021. [Online]. Available: http://themlbook.com/

[4] P. Ruiz Ruiz, “Intro Clustering.” http://www.pabloruizruiz10.com/resources/Curso-Machine-Learning-Esp/5---Aprendizaje-No-Supervisado/Intro-Clustering.html (accessed Sep. 23, 2021).

[5] Q. Wang and Z. Zhan, “Reinforcement Learning Model,Algorithms and Its Application,” 2011, Accessed: Sep. 23, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/6025669

[6] N. M. Mosharaf Kabir Chowdhury and R. Boutaba, “Network virtualization: State of the art and research challenges [Topics in Network and Service Management],” in *IEEE Communications Magazine*, 2009, vol. 47, no. 7, pp. 20–26. doi: 10.1109/MCOM.2009.5183468.

[7] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013, doi: 10.1109/SURV.2013.013013.00155.

[8] C. Haotong, W. Shengchen, H. Yue, L. Yun, and Y. Longxiang, “A_survey_of_embedding_algorithm_for_virtual_network_embedding,” 2019, Accessed: Oct. 09, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/8968720

[9] D. G. Andersen, “Theorical Approaches to Node Assignment,” 2002. Accessed: Sep. 29, 2021. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.119.1332

[10] “NP (Complexitat) - Viquipèdia, l’enciclopèdia lliure.” https://ca.wikipedia.org/wiki/NP_(Complexitat) (accessed Sep. 29, 2021).

[11] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. de Meer, “ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms,” vol. X, Jan. 2011, doi: 10.14279/tuj.eceasst.37.495.

[12] M. Duelli, D. Schlosser, J. F. Botero, X. Hesselbach, A. Fischer, and H. de Meer, “VNREAL: Virtual network resource embedding algorithms in the framework ALEVIN,” 2011. doi: 10.1109/NGI.2011.5985874.

[13] T. Canh Nguyen, “SARSA vs Q - learning.” https://tcnguyen.github.io/reinforcement_learning/sarsa_vs_q_learning.html (accessed Sep. 27, 2021).

[14] M. Duelli *et al.*, “Alevin2 / Wiki / home.” https://sourceforge.net/p/alevin/wiki/home/ (accessed Sep. 26, 2021).

[15] “Oracle VM VirtualBox.” https://www.virtualbox.org/ (accessed Sep. 26, 2021).

[16] “Enabling Open Innovation & Collaboration | The Eclipse Foundation.” https://www.eclipse.org/ (accessed Sep. 26, 2021).

[17] "JUNG - Java Universal Network/Graph Framework." http://jung.sourceforge.net/ (accessed Sep. 27, 2021).
[18] G. M. Parulkar, ACM Digital Library., and ACM Special Interest Group on Data Communication., *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures.* ACM, 2009.
[19] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," 2006. doi: 10.1109/INFOCOM.2006.322.
[20] X. Cheng *et al.*, "Virtual Network Embedding Through Topology-Aware Node Ranking," *ACM SIGCOMM*, vol. 41, 2011.
[21] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988, doi: 10.1109/49.12889.
[22] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A Theoretical Analysis of Deep Q-Learning," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.00137

Annex

Passos fets per poder executar ALEVIN 2.2

1. Descarregar el software de la web: https://sourceforge.net/projects/alevin/
2. Després de descomprimir el projecte afegir dintre la carpeta lib les llibreries necessàries (figura 31), ja que la descarrega automàtica dintre el projecte no funciona.
3. Descarregar el software que demana com a requeriments al README dintre del projecte.
4. Afegir a l'arxiu build.xml els jars afegits a la carpeta lib.
5. Compilar el projecte amb la comanda ANT
6. Executar ALEVIN.jar

Figura 31: Jars carpeta lib

Errors corregits interfície gràfica

No s'inicialitza el panell de la dreta (selecció i mapeig de links i nodes)

La versió 2.2 descarregada d'ALEVIN conté un error referent al panell de la dreta de la interfície gràfica on es mostren les seleccions de links i nodes i el mapeig de les VNRs sobre la xarxa física, aquest error, un cop executat l'algorisme no mostra res en el panell.

Investigant les classes que formen la interfície gràfica, concretament la principal que instancia els panells és veu que una part del constructor conté la inicialització del panell de la dreta, el que no està funcionant:

```
public GUI(Scenario scenario) {
        super("ALEVIN - ALgorithms for Embedding VIrtual Networks");

        // Pane initialization requires this.scenario to be set,
        // therefore it cannot take place in the super constructor.
        this.scenario = scenario;
        this.initializeRightPane();
        this.initializeCenterPane();



        singleton = this;

        .
        .
        .
```

En el codi s'inicialitza primer el panell de la dreta i després el panell central, mirant part de la construcció d'ambdós mètodes:

```
private void initializeRightPane() {
        JTabbedPane tabs = new JTabbedPane();

        tabs.addTab("Selection", new SelectionPanel(new
MySelectionTreeModel(),
                        graphpanel));
        tabs.addTab("Mapping",
                        mappingPanel = new MappingPanel(graphpanel,
scenario));

        tabs.setPreferredSize(new Dimension(250, 300));
        this.rightPane.add(tabs, BorderLayout.CENTER);
}
```

```
private void initializeCenterPane() {
            graphpanel = new MyGraphPanel(scenario);
            graphpanel.setPreferredSize(new Dimension(520, 300));
            graphpanel.setSynced(false);
            graphpanel.setLayerStack(scenario.getNetworkStack());

            graphpanel.addPropertyChangeListener(new
PropertyChangeListener() {
                  .
                  .
                  .
```

En el mètode que s'inicialitza el panell de la dreta es passa com a paràmetre l'atribut graphpanel, que aquest s'inicialitza al mètode del panell central, el que fa que quan inicialitzem primer el panell de la dreta aquest atribut no s'hagi inicialitzat i no és pugi inicialitzar el panell de la dreta. Per tant la solució més òptima és canviar l'ordre de la crida als mètodes quer inicialitzen els panells al constructor, cridant primer al que inicialitza el panell central i després el de la dreta.

No es veu la llista sencera de mètriques a la interfície gràfica

Aquest error no permet veure la llista sencera de mètriques a la interfície gràfica, ja que no surten totes en pantalla degut a que el menú desplegable no te cap barra ni fletxes fer pujar o baixar entre les mètriques. Per corregir aquest error simplement s'ha d'afegir al constructor de la classe MetricsMenu.java una barra o fletxa que es pugi moure entre les diferents mètriques.

Afegir algorisme interfície gràfica

Per afegir nous algorismes a la interfície gràfica s'han de seguir els següents passos:

1. Dintre el paquet vnreal.gui.dialog afegir una classe nomAlgorismeWizard.java, aquesta ha d'extendre de AbstractButtonDialog i ha d'implementar actionListener.

2. En aquesta classe s'ha d'implementar el panell que es vol mostrar a la interfície quan es clica a l'algorisme, amb els corresponents paràmetres de l'algorisme.

3. Un cop implementada la classe s'ha d'afegir al menu que conté la interfície de tots els algorismes, la classe AlgorithmsMenu.java al paquet vnreal.gui.Menu, inicialitzant la variable mi:

```
mi= new JMenuItem("Q learning");
mi.setActionCommand("Q learning");
mi.addActionListener(this);
add(mi);
```

4. Després s'ha d'afegir la seguent condició else if al mètode actionPerformed de la mateixa clase:

```
} else if (cmd.equals("Q learning")) {
      new QLearningWizard();
}
```

El resultat d'aquest procés és mostra a la figura 32.

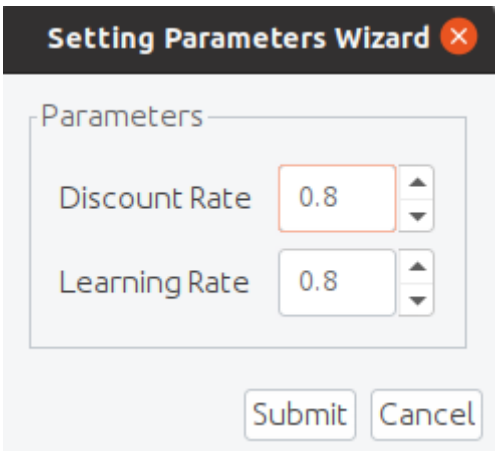

Figura 32: Panell algorisme Q learning

Glossari

VN - Virtual Network

NV - Network Virtualization

VNE - Virtual network embedding

SID - Subgraph isomorphism detection

ALEVIN – ALgorithms for Embedding VIrtual Networks

CPU – Central Processing Unit

InP – Infrastructure Provider

ISP – Internet Service Provider

VNR – Virtual Network Request

SVM - Support Vector Machine