Universitat de les Illes Balears

TREBALL DE FI DE GRAU

TÈCNIQUES DE COMPARACIÓ DE GRAFS: APLICACIÓ A LES XARXES METABÒLIQUES

Alba Linares Ferre

Grau de Matemàtiques

Escola Politècnica Superior

Any acadèmic 2023-24

TÈCNIQUES DE COMPARACIÓ DE GRAFS: APLICACIÓ A LES XARXES METABÒLIQUES

Alba Linares Ferre

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2023-24

Paraules clau del treball: grafs, kernels de grafs, xarxes metabòliques, comparació de grafs

Tutor: Maria de la Mercè LLabrés Segura

Vull agrair a na Mercè el temps i l'esforç dedicats a orientar-me en el desenvolupament d'aquest treball. També vull agrair el suport de la meva família i amics, especialment els meus pares.

SUMARI

Resum

Aquest treball s'emmarca en la branca de coneixement de la teoria de grafs, particularment en la comparació de grafs mitjançant els anomenats *kernels de grafs*. L'objectiu és entendre i explorar els diferents kernels de grafs existents i analitzar-ne la seva aplicació a la comparació de xarxes metabòliques, concretament als *m-DAGs* derivats dels *reaction grafs*. Els reaction grafs són grafs dirigits i etiquetats on els nodes representen reaccions metabòliques i les arestes, els compostos químics compartits. Els reaction grafs solen tenir mils de nodes, el que dificulta el seu estudi i, per això, s'han introduït els m-DAGs com els grafs quocients d'aquests per la relació d'accessibilitat mútua.

Primer, s'examinaran diversos tipus de kernels de grafs, seleccionant aquells més adequats pels *m-DAGs*. Els kernels de grafs escollits inclouen el *Node Histogram Kernel*, el *Shortest Path Kernel*, el *Weisfeiler-Lehman Kernel*, el *ODD-sth Kernel* i el *Pyramid Match Kernel*. Llavors, aquests seran aplicats a un conjunt de dades de 436 xarxes metabòliques per determinar quins són els més efectius. Per acabar, es proposa la creació d'un nou kernel combinant els que han mostrat millor rendiment, el que resulta en un kernel més eficient i que obté millors resultats.

CAPÍTOL 1

INTRODUCCIÓ

Aquest projecte s'emmarca en la branca de coneixement de teoria de grafs, en particular la comparació de grafs per mitjà dels anomenats *kernels de grafs*.

L'objectiu d'aquest treball és, d'una banda, entendre i explorar els diferents kernels de grafs introduïts a la literatura, i d'altra, analitzar-ne l'aplicació a la comparació de xarxes metabòliques. Cal dir que es denomina xarxa metabòlica al conjunt de processos metabòlics i físics que determinen la fisiologia i la bioquímica d'una cèl·lula. D'entre els models de xarxes metabòliques que s'han definit, aquí considerarem els anomenats *m-DAGs*. Els m-DAGs es defineixen a partir d'un altre model de xarxa metabòlica que són els *reaction grafs*, segons [1]. Els reaction grafs són grafs etiquetats i dirigits on els nodes representen les reaccions del metabolisme a analitzar i les arestes representen les relacions (composts químics compartits) entre aquestes reaccions. Ara bé, generalment es considera el metabolisme d'un organisme o el metabolisme generat per un conjunt de microorganismes, per la qual cosa el nombre de reaccions implicades és bastant elevat i, per tant, l'anàlisi dels reaction graphs és basant costós. Per aquest motiu, s'han introduït els m-DAGs (metabolic Directed Acyclic Graphs), que són els grafs obtinguts per mitjà de la condensació dels reaction grafs. Donat un graf dirigit, l'accessibilitat mútua de les parelles de nodes del graf és una relació d'equivalència, i, el graf quocient que s'obté per aquesta relació d'equivalència és el seu graf condensat. Intuïtivament, aquest graf quocient contreu en una classe d'equivalència tots els nodes que són mútuament accessibles, és a dir, que formen un cicle. El resultat, per tant, és, eliminar els cicles i generar un graf que pot arribar a ser molt més reduït.

Així doncs, aquest treball es centrarà primer en estudiar els diferents tipus de kernels de grafs existents [2, 3] i seleccionar aquells que millor s'adaptin a l'estructura dels m-DAGs. Llavors, aplicarem els kernels escollits a un data set de mostra format per 436 xarxes metabòliques de diferents espècies [4, 5] i, finalment, analitzarem els resultats per tal de poder decidir quin o quins kernels comparen millor els m-DAGs. Per acabar, crearem un nou kernel combinant els kernels que han donat bons resultats per tal de trobar un nou kernel més eficient.

Amb la finalitat de poder comparar grafs, s'han desenvolupat diferents algorismes

com els mètodes basats en l'isomorfisme de grafs, distàncies d'edició de grafs i descriptors topològics [2]. La popularització dels mètodes de kernels a l'aprenentatge automàtic i l'elevat cost computacional dels mètodes de comparació de grafs existents ha motivat el desenvolupament dels kernels de grafs. Abans de definir els kernels de grafs és necessari definir que és un kernel. Un kernel és una funció que mesura la similaritat entre dos objectes. Formalment, donat un conjunt X, un kernel és una funció $k : X \times X \to \mathbb{R}$ simètrica i definida positiva [6].

Els kernels de grafs difereixen en el tipus de subestructures que es tenen en compte a l'hora de definir la funció de kernel [2]. En concret, a aquest treball estudiarem els següents:

- El Node Histogram Kernel, que compara els histogrames de les etiquetes dels nodes dels grafs, proporcionant una mesura de similitud basada en les característiques dels nodes.
- El Shortest Path Kernel, que compara les distàncies dels camins més curts entre tots els parells de nodes dels grafs, és a dir, reflecteix la similitud en la connectivitat dels grafs.
- El Weisfeiler-Lehman Kernel, que per cada graf crea una seqüència de grafs i calcula les etiquetes dels nodes d'aquests de forma iterativa, combinant les etiquetes dels nodes amb les dels seus veïnats, capturant així estructures més complexes dels grafs.
- El ODD-sth Kernel, que descompon els grafs en grafs dirigits acíclics i mira quants de subarbres tenen en comú.
- El Pyramid Match kernel, que transforma els grafs en punts de l'espai $\mathbb{R}^d$ per calcular el kernel a partir d'aquests.

Veurem que el Node Histogram kernel, el kernel més senzill, obté resultats bastant bons pels m-DAGs del nostre data set, això es deu a que, com veurem més endavant, els m-DAGs tenen un nombre elevat de nodes aïllats. El Shortest Path i Weisfeiler Lehman són els kernels que han obtingut millors resultats a causa de l'estructura característica dels m-DAGs.

La memòria d'aquest TFG s'estructura de la següent manera. Al Capítol 2, el capítol de preliminars, introduïm els conceptes necessaris de Teoria de Grafs, definim formalment els Reaction graphs i m-DAGs i explicam breument els diferents mètodes de comparació de grafs que s'han creat fins a arribar als kernels, els quals veurem en detall.

Al Capítol 3, estudiarem (i donarem exemples detallats) 5 kernels en concret per ser els que millor s'ajusten a les característiques dels m-DAGs.

Al darrer capítol (Capítol 4), analitzarem la topologia dels m-DAGs del nostre data set, i, estudiarem els resultats obtinguts al aplicar els diferents kernels del Capítol 3 a aquests. Agruparem les dades aplicant un mètode de clustering a les matrius de kernels i avaluant els clústers obtinguts amb la classificació taxonòmica de les espècies del data set. Per acabar, combinarem els kernels que han obtingut millors resultats per crear un nou kernel.

CAPÍTOL 2

PRELIMINARS

En aquest capítol introduirem les definicions i notacions necessàries perquè aquest treball sigui el més autocontingut possible. Primer recordarem les nocions bàsiques de teoria de grafs (Secció 2.1), a continuació definirem alguns models de xarxes metabòliques (Secció 2.2) i, per acabar, introduirem breument les tècniques de comparació de grafs, especialment els anomenats *kernels* (Secció 2.3).

2.1 Teoria de Grafs

Un *graf* és un parell ordenat de conjunts $G = (V, E)$ on $V \neq \emptyset$ és el conjunt de nodes i E un conjunt de parells (ordenats o no) d'elements de V que és el conjunt d'arestes [7]. Direm que el graf és *dirigit* si $E \subseteq V \times V$, és a dir, si els parells de vèrtexs són ordenats. Altrament, direm que el graf és *no dirigit*. Denotarem una aresta $e \in E$ com $e = (u, v)$ si el graf és dirigit i com $e = uv$ en el cas no dirigit, amb el benentès que uv i vu són la mateixa aresta. Donat un graf G, donatarem per $V(G)$ el seu conjunt de nodes i per $E(G)$ el seu conjut d'arestes. Ara, donat un conjunt finit de grafs $\mathcal{G}$, denotarem el conjunt format per tots els nodes i totes les arestes de $\mathcal{G}$ com $V(\mathcal{G})$ i $E(\mathcal{G})$, respectivament.

Direm *ordre* d'un graf al cardinal del seu conjunt de nodes i *mida* d'un graf al cardinal del seu conjunt d'arestes.

En el cas dirigit, direm que l'aresta $e = (u, v)$ *uneix* u a v i anomenarem u el node inicial i v el node final. En el cas no dirigit, direm que l'aresta $e = uv$ *uneix* u i v i anomenarem aquests els extrems de l'aresta. Dos nodes són *adjacents* si existeix una aresta que els uneix. Una aresta és un *llaç* quan els extrems, o els nodes inicial i final, són el mateix node. Direm que un graf és *simple* si no té llaços.

Definim el *grau* d'un node d'un graf no dirigit $G = (V, E)$ com el nombre de nodes adjacents. Formalment,

$$d(v) = |\{u \in V | uv \in E, v \neq u\}|.$$

En el cas dirigit, definim el *grau d'entrada* d'un node com el nombre d'arestes que

tenen el node com a node final. Formalment,

$$d_e(v) = |\{u \in V | (u,v) \in E, v \neq u\}|,$$

i definim el *grau de sortida* d'un node com el nombre d'arestes que tenen el node com a node inicial. Formalment,

$$d_s(v) = |\{u \in V | (v,u) \in E, v \neq u\}|.$$

Definirem un *graf amb etiquetes als nodes* com un graf $G = (V,E)$ que té una sola etiqueta o un sol vector associat a cada node mitjançant la funció $l : V \to L$, on L és el conjunt d'etiquetes, o $f : V \to \mathbb{R}^d$, respectivament. Cal mencionar també el *graf etiquetat o atribut a les arestes* que té assignades etiquetes o atributs a les arestes. D'ara en endavant, direm *graf etiquetat o atribuït* al *graf amb etiquetes als nodes*.

Un *graf ponderat o amb pesos* és un graf $G = (V,E)$ que té una funció $w : E \to R$ que assigna pesos a les arestes.

Donat un graf dirigit G i v un node d'aquest, definim el seu *vecindari* com el conjunt de nodes que estan connectats a aquest mitjançant una aresta, és a dir, $N_G(v) = \{u \in V | (v,u) \in E\}$. Es defineix de forma anàloga al cas no dirigit.

Definim la *matriu d'adjacència* d'un graf dirigit G com una matriu quadrada $A = (a_{u,v})$ de dimensió $n = |V|$ on

$$a_{u,v} = \begin{cases} 1 & \text{si } (u,v) \in E \\ \\ 0 & \text{altrament} \end{cases}$$

Si, a més, G és ponderat i $w : E \to \mathbb{R}$ és la funció que assigna pesos a les arestes, aleshores la seva matriu d'adjacència és

$$a_{u,v} = \begin{cases} w((u,v)) & \text{si } (u,v) \in E \\ \\ 0 & \text{altrament} \end{cases}$$

Notem que en el cas no dirigit es fa de forma anàloga i tendrem una matriu simètrica. Un *isomorfisme* entre dos grafs $G = (V,E)$ i $G' = (V',E')$ és una bijecció dels nodes d'un graf sobre l'altre de tal forma que es preservi l'adjacència dels nodes, és a dir, una bijecció $f : V(G) \to V(G')$ tal que $(u,v) \in E$ si, i només si, $(f(u), f(v)) \in E'$, $\forall u, v \in V$.

Un *invariant* per la classe d'isomorfisme és una propietat que es conserva per isomorfisme, és a dir, si un graf té tal propietat, els grafs isomorfs a ell també la tendran. Per tant, els invariants estableixen condicions necessàries, però no suficients, d'isomorfisme.

Un graf *bipartit* és un graf no dirigit tal que el conjunt de nodes es pot dividir en dos conjunts disjunts tal que no hi hagi arestes entre nodes del mateix conjunt. Formalment, $G = (V,E)$ és bipartit si existeixen dos conjunts A i B tals que $V = A \cup B$, $A \cap B = \emptyset$ i $uv \in E \to (u \in A \wedge v \in B) \oplus (u \in B \wedge v \in A)$.

Ara, direm que un graf bipartit G, amb bipartició (A,B), és *complet* quan hi ha una aresta entre parells de nodes de A i B.

A continuació, introduirem el concepte de *Maximum Weight Matching* a grafs bipartits.

Donat un graf $G = (V, E)$ bipartit complet i ponderat, amb bipartició (A, B) i funció de pesos $w : E \to \mathbb{R}$, existeix un subgraf M de G tal que cada node de A i B està connectat per només una aresta, anomenat matching, de manera que maximitza la funció

$$w(M) = \sum_{e \in M} w(e).$$

Sabem que sempre existeix aquest matching M i es pot trobar mitjançant el conegut Munkres o Hungarian algorithm [8].

Sigui $G = (V, E)$ un graf i siguin $u, v \in V$. Un *camí* dins G de u a v és una seqüència de nodes $(v_0, v_1, ..., v_k)$ tal que:

- $u = v_0$ i $v = v_k$
- $(v_i, v_{i+1}) \in E \ \forall i \in \{0, ..., k-1\}$
- $v_i \neq v_j \ \forall i, j \in \{0, ..., k-1\}$

De manera anàloga definim un camí dins un graf G no dirigit canviant $(v_i, v_{i+1}) \in E$ per $v_i v_{i+1} \in E \ \forall i \in \{0, ..., k-1\}$.

Sigui $(v_0, v_1, ..., v_k)$ la seqüència de nodes que defineix un camí a un graf $G = (V, E)$, definim la *distància* d'aquest camí com el nombre d'arestes que formen aquest camí, és a dir, $k-1$.

Definim un *cicle* com un camí on el primer node coincideix amb el darrer.

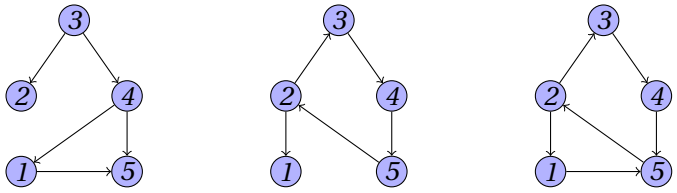
Donat un graf dirigit G, s'anomena *graf no dirigit subjacent a* G al graf que s'obté en suprimir l'orientació de les arestes; és a dir, té els mateixos nodes que G i dos nodes u, v estan units per una aresta si (u, v) o (v, u) és una aresta de G.

Un node v es diu que és *accessible* des d'un node u si existeix un camí de u a v; dos nodes u, v es diu que són *mútuament accessibles* si existeixen camins de u a v i de v a u. La relació d'accessibilitat mútua (dos nodes estan relacionats si són mútuament accessibles) és una relació d'equivalència.

El fet que els camins no siguin reversibles (perquè tractam grafs dirigits) fa que apareguin diferents nocions de connexitat. Un graf és *dèbilment connex* si el seu graf no dirigit subjacent és connex, *unilateralment connex* si per a tot parell de nodes u, v, o bé v és accessible des de u o bé u és accessible des de v i *fortament connex* si cada parell de nodes són mútuament accessibles.

Observem que la connexitat forta implica la unilateral, i aquesta implica la dèbil.

Exemple 1 *Vegem un exemple dels diferents tipus de connexitat:*



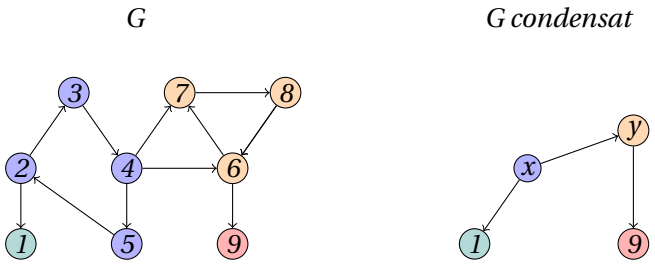
Dèbilment connex Unilateralment connex Fortament connex

Notem que en el primer graf no existeix cap camí del node 2 al node 4 ni del node 4 al node 2 i, per tant, no és unilateralment connex ni fortament connex. Vegem també que en el segon graf no existeix cap camí del node 1 al 2 i, per tant, no és fortament connex.

Sigui un graf, anomenarem *components connexes* del graf als subgrafs del graf tals que els seus nodes no tenen connexions amb nodes de fora d'aquest subconjunt i cada un dels quals és connex. Notem que si el graf és connex, aquest té una sola component connexa.

Donat un graf dirigit $G = (V, E)$, i una relació d'equivalència $\sim$ a $V \times V$, definim el graf quocient $G/\sim = (V/\sim, E/\sim)$ com el graf que té per nodes les classes d'equivalència mòdul $\sim$ i $([u]_\sim, [v]_\sim) \in E/\sim$ si existeixen $u' \in [u]_\sim$ i $v' \in [v]_\sim$ tals que $(u', v') \in E$. Definim el *graf condensat* de G com el seu graf quocient respecte de la relació d'accessibilitat mútua.

Exemple 2 *A la figura següent apareix un graf i el seu graf condensat.*



Notem que els subgrafs fortament connexos de G es transformen en nodes del graf condensat. En aquest exemple, notem que els nodes 2, 3, 4 i 5 són mútuament accessibles entre ells i, per tant, es condensen en un sol node en el graf condensat, el node x. De la mateixa forma, els nodes 6, 7 i 8 es condensen amb un sol node, el node y, en el graf condensat.

Distingim un tipus especial de grafs dirigits, els *grafs dirigits acíclics* o DAGs (de l'anglès directed acyclic graph), caracteritzats pel fet que no tenen cicles.

Proposició 3 *El graf condensat d'un graf dirigit qualsevol és un DAG.*

Prova.- Suposem que existeix un cicle $[u_0], [u_1], [u_2], \ldots, [u_k] = [u_0]$ al graf condensat. Aleshores existeixen representants $v_i, w_i \in [u_i]$ tals que $v_i w_{i+1}$ és una aresta de G. Com els nodes v_i i w_i pertanyen a la mateixa classe existeixen camins de w_i a v_i; concatenant les arestes i camins trobats, s'arriba a una seqüència de nodes $u_0, \ldots, v_0, w_1, \ldots, v_1, w_2, \ldots, v_k$,
$w_k, \ldots, u_0$, cosa que implica que tots els nodes trobats són mútuament accessibles. Per tant, $[u_0] = [u_1] = \cdots = [u_k]$ i no existeix cap cicle al graf condensat. □

Donat un graf dirigit, anomenam *arrel* a un node que no té arestes entrants, és a dir, el grau d'entrada val zero i anomenam *fulla* a un node que no té arestes sortints, és a dir, el grau de sortida val zero. Un *arbre* és un graf connex, no dirigit i sense cicles. Un arbre *arrelat* és un arbre on consideram un node distingit que anomenem arrel (es pot considerar com un DAG, considerant la direcció de les arestes en el sentit de les fulles). A aquest treball quan xerrem d'arbres ens referirem sempre a arbres arrelats.

D'ara en endavant, considerem que tots els grafs (dirigits o no) són simples. Si no es menciona el contrari, suposarem que el grafs són dirigits i que no tenen llaços.

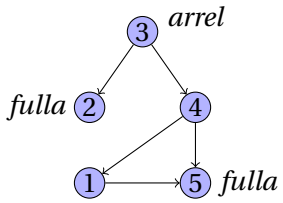


Figura 2.1: Vegem un exemple de graf dirigit

2.2 Xarxes metabòliques

Una xarxa metabòlica representa el conjunt de processos metabòlics i físics que determinen les propietats fisiològiques i bioquímiques d'una cèl·lula. Introduïm aquí dos models de xarxes metabòliques, els *reaction graphs* i els *metabolic DAGs* que s'han definit a la literatura per analitzar les xarxes metabòliques [1].

Reaction Graphs

En aquest model de xarxa metabòlica es representen les reaccions químiques com a nodes i els camins metabòlics com les interaccions (arestes) que guien aquestes reaccions. Les reaccions químiques succeeixen quan es trenquen o es formen enllaços químics entre els àtoms. Les substàncies que participen en una reacció química es coneixen com els metabòlits, els metabòlits que la reacció transforma s'anomenen substrats i els metabòlits que es produeixen al final de la reacció s'anomenen productes. Els enzims són proteïnes que tenen com a funció accelerar la velocitat a la qual és produeix una certa reacció. Aquesta funció s'anomena de catalització.

Formalment, definirem una reacció química com una terna $R_i = (I_i, E_i, O_i)$, on I_i és el conjunt dels substrats, E_i l'enzim que catalitza la reacció i O_i el conjunt format pels productes. Aleshores, definim un *reaction graph* com un graf dirigit $G_R = (R, E)$ on el conjunt de nodes és un conjunt de reaccions químiques i $(R_i, R_j) \in E$ si algun producte de R_i és també un substrat de R_j, és a dir, el conjunt d'arestes és

$$E = \{(R_i, R_j) | \exists c \in O_i \cap I_j\}.$$

Notem que els metabòlits no es veuen representats al reaction graph, però es tenen en compte a l'hora de definir les arestes.

Una reacció es diu que és *reversible* quan pot ocórrer en els dos sentits, és a dir, dels substrats als productes o dels productes als substrats. Diferents factors, com son el tipus de reacció, la concentració dels metabòlits, la temperatura o la pressió poden condicionar que una reacció sigui reversible o no. En el model dels reaction grafs, es consideren dos nodes diferents per representar les reaccions reversibles.

Molt sovint, per tal d'analitzar el metabolisme d'un organisme, es considera la xarxa metabòlica que conté totes les seves reaccions. En aquest cas, el Reaction Graph que s'obté té milers de nodes (reaccions) i, per tant, els Reaction Graphs solen ser grafs d'ordre i mida molt gran amb milers de nodes i arestes que sovint dificulten el seu estudi. Aquest fet és el que motivà la definició dels metabòlic DAGs. A la Figura 2.2 mostram el Reaction Graph del Oryctolagus cuniculus (conill) generat a partir de la informació metabòlica de la base de dades KEGG [4, 5].

Figura 2.2: Reaction Graph del Oryctolagus cuniculus (conill)

Metabolic DAGs

Per facilitar l'estudi de les xarxes metabòliques, s'han introduït els anomenats metabolic DAGs que s'obtenen com el graf condensat d'un Reaction Graph.

Notem que per a cada Reaction Graph G_R podem considerar la seva col·lecció de components fortament connexes, anomenats Metabolic building blocs (MBB), i calcular el graf condensat d'aquest que, com hem vist anteriorment, serà un DAG. Anomenarem metabolic DAG o m-DAG de forma abreujada a aquest nou graf.

La idea que hi ha darrere de considerar el graf condensat és que els MBB són subgrafs tals que existeix un camí en els dos sentits entre tots els seus nodes i es poden pensar com a subgrafs compactes dins el Reacton graph. En canvi, els nodes que no són mútuament accessibles amb cap altre node simbolitzen reaccions que són necessàries per a la connectivitat de la xarxa metabòlica. En conclusió, el graf condensat ens dona informació sobre la connectivitat de la xarxa. A la Figura 2.3 mostram el m-DAG del Oryctolagus cuniculus (conill) obtingut a partir del reaction graph mitjançant l'eina [9].

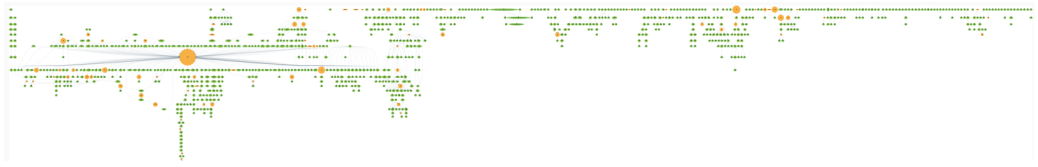


Figura 2.3: En aquesta figura mostram m-DAG obtingut del reaction graph de la Figura 2.2

2.3 Comparació de grafs

Abans dels kernels es van emprar altres algoritmes per comparar grafs com els mètodes basats en l'isomorfisme de grafs, distàncies d'edició de grafs i descriptors topològics [2].

La definició d'isomorfisme entre grafs dona lloc a una relació d'equivalència que permet separar un conjunt de grafs en classes d'equivalència diferents on els grafs que pertanyen a la mateixa classe són indistingibles uns dels altres. Aquest fet permet definir una mesura entre grafs. El cost computacional dels algorismes per resoldre el problema de l'isomorfisme de grafs és bastant elevat, llevat dels grafs que són arbres, no es coneix cap algorisme en temps polinomial per resoldre aquest problema. Per aquest motiu i perquè aquests algorismes són bastant restrictius, es varen desenvolupar altres mètodes per la comparació de grafs.

Un d'aquests mètodes és la distància d'edició de grafs. Donats dos grafs G i G' la *distància d'edició de grafs* mesura quantes transformacions són necessàries per a convertir G en G' tenint en compte que cada tipus transformació (intersecció, eliminació o substitució de nodes o arestes, etc.) té un pes. En concret, la distància d'edició de grafs és la suma dels costs del conjunt de transformacions que són necessàries per transformar G en G' de menys pes. La flexibilitat de la definició de la funció de cost permet adaptar-la al tipus de grafs, però trobar una bona parametrització de la funció de cost és complicat. Una vegada la funció de cost està fixada, existeixen molts d'algorismes per calcular la distància d'edició entre dos grafs, però de la mateixa forma que als algoritmes d'isomorfisme, el seu cost computacional és molt alt.

Una altra família de mètodes per a comparar grafs és la que considera descriptors topològics i invariants de grafs. La idea principal és representar un graf com un vector. Els descriptors topològics són representacions vectorials de propietats topològiques. Malauradament, resoldre el problema dels grafs invariants equival computacionalment a resoldre el problema d'isomorfisme de grafs. Un altre dels descriptors més usats és el càlcul dels vectors i valors propis de la matriu Laplaciana del graf.

Amb la finalitat de millorar l'eficiència i l'expressivitat dels mètodes anteriors es van proposar els kernels de grafs. Aquests consideren les seves característiques topològiques (graus de nodes, connectivitat, distàncies entre nodes, cliques, etc.) per comparar-los. El kernels el que fan és traduir les característiques topològiques a un vector de manera que comparar grafs es redueix a comparar vectors. Per comparar vectors, una manera fàcil és fer-ne el seu producte escalar, i per això es consideren els espais de Hilbert, que són els espais vectorials normats i complets.

2.3.1 Kernels

Definirem el kernel entre dos objectes com una funció que mesura la similitud entre ells.

Abans de definir formalment el terme de kernel, cal definir el concepte de funció simètrica i (semi)definida positiva. Sigui una funció $f : X \times X \to \mathbb{R}$ on X és un conjunt no buit. Direm que f és *simètrica* si satisfà $f(x,x') = f(x',x)\ \forall x,x' \in X$. Direm que f és *semidefinida positiva* si $\forall \lambda_1,...,\lambda_k \in \mathbb{R}$ i $\forall x_1,...x_k \in X$ tenim

$$\sum_{i=1}^{k}\sum_{j=1}^{k} \lambda_i \lambda_j f(x_i,x_j) \geq 0$$

(*estrictament definida positiva* si la igualtat s'assoleix només quan tots els λ_i valen zero per qualssevol valors dels x_i). Equivalentment, una matriu associada A a una aplicació $f : X \times X \to \mathbb{R}$ és (semi)definida positiva si tots els seus valors propis són positius (positius o nuls).

A continuació, donam la definició de kernel d'acord a [6].

Definició 4 (Kernel) *Considerem X un conjunt no buit una funció $K : X \times X \to \mathbb{R}$ direm que és un* kernel *a $X \times X$ si K és una funció simètrica i semidefinida positiva, és a dir, $\forall x, y \in X\ K(x,y) = K(y,x)$ i $\forall n \geq 1$ i $x_1, ..., x_n \in X$ la matriu K definida per $K_{ij} = K(x_i, x_j)$ és semidefinida positiva.*

El *(semi)producte escalar* és una aplicació $\langle \cdot, \cdot \rangle : X \times X \to \mathbb{R}$ bilineal, simètrica i (semi)definida positiva.

Proposició 5 *Donat $x \in X$ i donada la funció de mapeig $\varphi(x) = \{\varphi_n(x)\}_{n \geq 1}$ que assigna una seqüència de nombres reals a cada $x \in X$, si $K : X \times X \to \mathbb{R}$ definida per*

$$K(x, x') = \sum_n \phi_n(x)\phi_n(y) = \langle \phi(x), \phi(x') \rangle$$

per tot $x, x' \in X$ és un producte escalar, aleshores K és un kernel.

Prova.

Notem que l'aplicació K és simètrica per ser el producte escalar i semidefinida positiva perquè per tot $x_1, ..., x_n \in X$ i per tot $\lambda_1, ..., \lambda_n \in \mathbb{R}$ es té

$$\sum_{i,j=1}^{n} \lambda_i \lambda_j K(x_i, x_j) = \sum_{i,j=1}^{n} \lambda_i \lambda_j \langle \phi(x_i), \phi(x_j) \rangle =$$

$$\sum_{i=1}^{n} \left\langle \lambda_i \phi(x_i), \sum_{j=1}^{n} \lambda_j \phi(x_j) \right\rangle = \left\langle \sum_{i=1}^{n} \lambda_i \phi(x_i), \sum_{j=1}^{n} \lambda_j \phi(x_j) \right\rangle =$$

$$\left\langle \sum_{i=1}^{n} \lambda_i \phi(x_i), \sum_{i=1}^{n} \lambda_i \phi(x_i) \right\rangle \geq 0$$

per ser el producte escalar bilineal i semidefinida positiva. □

Vegem un kernel bàsic per la comparació de grafs. El *Node Kernel* que compara les etiquetes de dos nodes.

Definició 6 (Node Kernel) *Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats (no necessàriament del mateix ordre) i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt de les etiquetes. Per a cada $G = (V, E) \in \mathcal{G}$ amb $l : V \to L$ la funció que assigna una etiqueta a cada node del graf. Donats $G, G' \in \mathcal{G}$ i siguin $u, v \in V(\mathcal{G})$ dos nodes qualssevol, consideram l'anomenat* Node Kernel *$K_{node} : V(\mathcal{G}) \times V(\mathcal{G}) \to \mathbb{R}$ definit per*

$$k_{node}(u, v) = \begin{cases} 1 & \text{si } l(u) = l'(v) \\ 0 & \text{altrament} \end{cases}$$

La idea es basa en la funció delta de Dirac, és a dir, val 1 si dos nodes tenen la mateixa etiqueta i val 0 en cas contrari.

De forma anàloga, podem definir el *Edge Kernel*, K_{edge}, com un kernel que compara les etiquetes o pesos de dos arestes.

Proposició 7 *El Node Kernel és un kernel.*

Prova. Notem que podem reescriure el kernel com $k(u,v) = \langle \phi(u), \phi(v) \rangle$ on $\phi : V(\mathcal{G}) \to \{0,1\}^d$ tal que per un $u \in V(\mathcal{G})$ qualsevol, $\phi(u)$ és un vector de longitud d definit per

$$\phi_i(u) = \begin{cases} 1 & \text{si } l(u) = i \\ \\ 0 & \text{altrament} \end{cases}$$

i, per tant, és un kernel. □

Ara, volem veure la implicació en sentit contrari (Teorema 11), és a dir, que donat un kernel K existeix una funció ϕ tal que $K(x,x') = \langle \phi(x), \phi(x') \rangle$ per tot $x, x' \in X$. Abans, definirem alguns conceptes previs que són necessaris.

Definició 8 (Espai de Hilbert) *Un espai de Hilbert és un espai vectorial H definit sobre un cos K que té una norma associada $||\cdot|| = \sqrt{\langle \cdot, \cdot \rangle}$ i és complet respecte d'aquesta norma, és a dir, cada successió de Cauchy de H convergeix a un element de H.*

Exemple 9 *Vegem que $\mathbb{R}^n$, amb la norma associada $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^n x_i^2}$, és un espai de Hilbert.*

Vegem que és complet, és a dir, cada successió de Cauchy en $\mathbb{R}^n$ convergeix a un element de $\mathbb{R}^n$. Donada $\{\mathbf{x}^{(k)}\} \subseteq \mathbb{R}^n$ una successió de Cauchy en $\mathbb{R}^n$, per tot $\epsilon > 0$, existeix un $N \in \mathbb{N}$ tal que per tot $k, l \geq N$, $\|\mathbf{x}^{(k)} - \mathbf{x}^{(l)}\| < \epsilon$. Això significa que les components individuals de la successió són successions de Cauchy en $\mathbb{R}$, és a dir, si $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$, llavors per cada $i \in \{1, \ldots, n\}$ la successió $\{x_i^{(k)}\}$ és una successió de Cauchy en $\mathbb{R}$. Ara per ser $\mathbb{R}$ complet tenim que existeix un límit a $\mathbb{R}$ per cada una de les successions anteriors i, per tant, existeix un vector límit $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ a $\mathbb{R}^n$ de la successió inicial $\{\mathbf{x}^{(k)}\} \subseteq \mathbb{R}^n$.

Definició 10 (Kernel Reproductor i Espai de Hilbert Kernel Reproductor) *Un* Espai de Hilbert Kernel Reproductor *sobre un conjunt no buit X és un espai de Hilbert H de funcions $f : X \to \mathbb{R}$ amb un kernel $K : X \times X \to \mathbb{R}$ tal que*

i) $K(\cdot, x) \in H$ *per tot* $x \in X$,

ii) $f(x) = \langle f, K(\cdot, x) \rangle$ *per tot* $f \in H$ *i* $x \in X$.

Un kernel que satisfà les propietats anteriors s'anomena un Kernel Reproductor.

El *Teorema de Moore–Aronszajn*, un dels teoremes claus en la teoria de kernels, estableix que qualsevol kernel defineix un únic Espai de Hilbert reproductor [10]. No donarem aquí la demostració d'aquest teorema, però sí la del teorema següent que ens servirà per poder demostrar que els kernels de grafs del capítol 3 són, efectivament, kernels.

Teorema 11 *Sigui X un espai mètric separable, aleshores per a qualsevol kernel $K : X \times X \to \mathbb{R}$ continu, l'Espai de Hilbert Kernel Reproductor associat H és separable. A més, en aquest cas, H te una base ortonormal $\{\phi_n\}_{n\geq 1}$ tal que per qualssevol $x, y \in X$,*

$$K(x,y) = \sum_n \phi_n(x)\phi_n(y).$$

Prova. Donarem un esquema de la demostració, es pot trobar la prova completa a [6].

Abans, introduirem alguns conceptes. Sigui un conjunt X dotat d'una mètrica $d(x,y)$. Donada una funció $f : X^n \to \mathbb{R}$ per $n \geq 1$, direm que f és *contínua* si per tot $\epsilon > 0$ existeix $\delta > 0$ tal que si es satisfà $d(x_i, y_i) \leq \delta$ per tot $1 \leq i \leq n$ aleshores $|f(x_1,\ldots,x_n) - f(y_1,\ldots,y_n)| \leq \epsilon$. Direm que un conjunt X és *separable* si existeix un conjunt numerable $X_0 \subseteq X$ tal que per tot $x \in X$ i per tot $\epsilon \geq 0$ existeix $y \in X_0$ tal que $d(x,y) \leq \epsilon$.

Ara sí, donam un esquema de la demostracó del teorema. Donat X un espai separable i $K : X \times X \to \mathbb{R}$ una funció continua. Considerem H_0 l'espai de funcions lineals de X en $\mathbb{R}$ generat per les funcions $\{K_x : x \in X\}$ on $K_x(y) = K(x,y)$.

Siguin $f = \sum_{i=1}^n \lambda_i K_{x_i}$ i $g = \sum_{j=1}^m \alpha_j K_{y_j}$ de H_0 on $\lambda_i, \alpha_i \in \mathbb{R}$ per $1 \leq i \leq n$ i $1 \leq j \leq m$, definim

$$\langle f, g\rangle = \sum_{j=1}^m \alpha_j f(y_j) = \sum_{i,j} \lambda_i \alpha_j K(x_i, y_j) = \sum_{i=1}^n \lambda_i f(x_i).$$

Es pot veure que hem definit un semiproducte escalar a H_0 i que és un kernel que satisfà la propietat reproductora. Llavors, per ser un kernel a $H_0 \times H_0$, usant la desigualtat de Cauchy Schwartz generalitzada per Kernels, tenim que $\langle\cdot,\cdot\rangle$ és un producte escalar en H_0. Notem que podem completar H_0 i el seu producte escalar a un espai H que conté les funcions de H_0 i és un espai de Hilbert. Llavors, H és un *Espai de Hilbert Kernel Reproductor*. A més, es pot demostrar que H és separable ja que X és un espai mètric separable i K és continua i, per tant, té una base numerable ortonormal $\{\phi_n\}_{n\geq 1}$. Finalment, com H conté H_0, podem escriure

$$K_x(y) = \sum_n \langle K_x, \phi_n\rangle \phi_n(y) = \sum_n \phi_n(x)\phi_n(y)$$

on la segona igualtat es satisfà per la propietat reproductora.

□

A partir d'aquest resultat podem deduir el lema següent.

Lema 12 *Sigui K un kernel a un conjunt $X \times X$ i donats $A, B \subseteq X$ dos conjunts finits i no buits, definim $K'(A,B) = \sum_{x\in A, y\in B} K(x,y)$. Aleshores, si $\mathcal{P}(X)$ és el conjunt format per tots els subconjunts finits i no buits de X, K' és un kernel a $\mathcal{P}(X) \times \mathcal{P}(X)$.*

Prova. Per qualsevol subconjunt finit $A \subseteq X$, considerem $f_a = \sum_{x\in A} K_x \in H_0$ on H_0 és l'espai pre-Hilbert associat a K, de forma anàloga a la demostració del teorema 11. Si per $A, B \subseteq X$ dos subconjunts finits no buits, definim $K'(A,B) = \sum_{x\in A, y\in B} K(x,y)$ aleshores $K'(A,B) = \langle f_A, f_B\rangle$. Llavors, per ser $\langle\cdot,\cdot\rangle$ un kernel, tenim que K' és un kernel a $\mathcal{P}(X) \times \mathcal{P}(X)$. □

Vegem ara algunes propietats dels kernels.

Proposició 13 *Donats dos kernels K_1 i K_2 a $X \times X$, es satisfan les propietats següents:*

(i) *λK_1 amb $\alpha \in \mathbb{R}^+$ és un kernel a $X \times X$.*

(ii) *$K_1 + K_2$ és un kernel a $X \times X$.*

(iii) *$K_1 \cdot K_2$ és un kernel a $X \times X$.*

Prova. Vegem que satisfan la definició de kernel. (i) Sabem que el producte per un escalar amb una funció simètrica és simètrica i notem que per tot $x_1, ..., x_n \in X$ i per tot $\lambda_1, ..., \lambda_n \in \mathbb{R}$ tenim $\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j (\alpha K_1(x_i, x_j)) = \alpha \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j K_1(x_i, x_j) \geq 0$ per ser K_1 un kernel a $X \times X$ i $\alpha > 0$.

(ii) Sabem que la suma de dues funcions simètriques és simètrica i notem que per tot $x_1, ..., x_n \in X$ i per tot $\lambda_1, ..., \lambda_n \in \mathbb{R}$, tenim

$$\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j (K_1(x_i, x_j) + K_2(x_i, x_j)) = \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j K_1(x_i, x_j) + \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j K_2(x_i, x_j) \geq 0$$

per ser K_1 i K_2 kernels a $X \times X$.

(iii) Sabem que el producte de dues funcions simètriques és simètrica. Amb l'objectiu de provar que el producte de funcions semidefinides positives és una funció semidefinida positiva és defineix el *producte de Hadamard*. El producte de Hadamard de dues matrius (de la mateixa dimensió) és el resultat de fer el producte component a component. Notem que el problema es redueix a provar que el producte de Hadamard de dues matrius semidefinides positives és una matriu semidefinida positiva. Podeu consultar la prova a [11]. □

Del resultat anterior es pot deduir el següent.

Proposició 14 *Donats els kernels K_1 a $X_1 \times X_1$ i K_2 a $X_2 \times X_2$ i siguin $(a, c) \in X_1 \times X_1$, $(b, d) \in X_2 \times X_2$ aleshores es satisfan les propietats següents:*

(i) *$K_1 \otimes K_2((a, b), (c, d)) = K_1(a, c)K_2(b, d)$ és un kernel a $(X_1 \times X_2) \times (X_1 \times X_2)$.*

(ii) *$K_1 \oplus K_2((a, b), (c, d)) = K_1(a, c) + K_2(b, d)$ és un kernel a $(X_1 \times X_2) \times (X_1 \times X_2)$.*

Prova. Notem que les funcions de (i) i (ii) són simètriques i semidefinides positives de forma anàloga a la prova de la Proposició 13. Llavors són kernels a $(X_1 \times X_2) \times (X_1 \times X_2)$. □

Definició 15 (Zero extensió) *Si $S \subseteq X$ i K és un kernel a $S \times S$ aleshores podem estendre K a un kernel de $X \times X$, anomenat zero extensió de K, definint $K(x, y) = 0$ quan x o y no pertany a S.*

La zero extensió d'un kernel és un kernel per ser K simètric i semidefinit positiu.

Notem que al revés també passa, és a dir, si K és un kernel a $X \times X$, K restringit a $S \times S$, on $S \subset X$, és un kernel perquè, és fàcil veure que segueix essent simètric i semidefinit positiu.

A continuació, definirem un kernel que compara les etiquetes o pesos de dues arestes juntament amb les etiquetes dels seus nodes extrems.

Definició 16 (Path Kernel) *Sigui $\mathcal{G}$ un conjunt finit de grafs dirigits amb etiquetes als nodes i etiquetes o pesos a les arestes, donades dues arestes $(u,v),(u',v') \in E(\mathcal{G})$ definim el Path Kernel a $E(\mathcal{G}) \times E(\mathcal{G})$ com*

$$K_{path}((u,v),(u',v')) = K_{node}(u,u')K_{edge}((u,v),(u',v'))K_{node}(v,v').$$

En el cas de grafs no dirigits, es pot definir el path kernel considerant els grafs dirigits amb arestes en els dos sentits.

Observem que donades dues arestes $e = (u,v)$ i $e' = (u',v')$ de $E(\mathcal{G})$, $k_{path}(e,e') = 1$ quan $l(u) = l(u')$, $l(v) = l(v')$ i el pes de les arestes e i e' coincideix.

Proposició 17 *El Path Kernel és un kernel.*

Prova. Notem que $K_{node'}((u,v),(u',v')) = K_{node}(u,u')K_{node}(v,v')$ és un kernel a $(V(\mathcal{G}) \times V(\mathcal{G})) \times (V(\mathcal{G}) \times V(\mathcal{G}))$ per la Proposició 14. Ara, com $E(\mathcal{G}) \subseteq (V(\mathcal{G}) \times V(\mathcal{G}))$, podem estendre (Definició 15) el kernel K_{edge} a $(V(\mathcal{G}) \times V(\mathcal{G})) \times (V(\mathcal{G}) \times V(\mathcal{G}))$. Per tant, $K_{path} = K_{node}(u,u')K_{node}(v,v')K_{edge}((u,v),(u',v')) = K_{node'}((u,v),(u',v'))K_{edge}((u,v),(u',v'))$ és un kernel a $(V(\mathcal{G}) \times V(\mathcal{G})) \times (V(\mathcal{G}) \times V(\mathcal{G}))$ per la Proposició 14. Finalment, podem restringir el kernel K_{path} a $E(\mathcal{G}) \times E(\mathcal{G})$.

□

En aquest treball consideram els kernels com a mesures de similitud de grafs, per tant, X és un conjunt finit de grafs. Així doncs, definir un kernel en alguns casos es redueix a definir una aplicació $\phi : X \to \mathbb{R}^n$ que a cada graf $G \in X$ li assigna un vector de $\mathbb{R}^n$, i el kernel de dos grafs, $K(G,G')$, és el producte escalar dels vectors associats als grafs, $\langle \phi(G), \phi(G') \rangle$.

D'altra banda, també ens interessa proporcionar un algoritme per obtenir kernels de comparació grafs basats en la descomposició en subestructures d'aquests grafs. Per això es van desenvolupar els anomenats kernels R-convolucionals [6].

Definició 18 (R-descomposició d'un graf) *Donada una família de grafs $\mathcal{G}$, definim la* R-descomposició *del graf $G \in \mathcal{G}$ com $R(g_1,...,g_d,G)$ on $g_i \in G$ és una part de G (un subgraf, un subconjunt de nodes de G, etc.). La* preimatge *o* fibra *d'una R-descomposició és*

$$R^{-1}(G) = \{\hat{g} = (g_1,...,g_d) | R(g_1,...,g_d,G)\}.$$

A continuació veurem que l'existència de kernels per comparar les parts g_i de G suposa l'existència de kernels a $\mathcal{G}$.

Definició 19 (Kernel R-convolucional) *Sigui $i \in \{1,...,d\}$, K_i el kernel de les parts $\mathcal{G}$. Definim el kernel R-convolucional entre dos grafs $G, G' \in \mathcal{G}$ com*

$$K_R(G,G') = \sum_{\hat{g} \in R^{-1}(G)} \sum_{\hat{g}' \in R^{-1}(G')} \prod_{i=1}^{d} K_i(g_i, g_i').$$

Proposició 20 *El kernel R-convolucional és un kernel a $\mathcal{G} \times \mathcal{G}$*

Prova. Sabem que $\hat{K}(\hat{g},\hat{g'}) = \prod_{i=1}^{d} K_i(g_i, g_i')$ és un kernel a $\mathcal{G} \times \mathcal{G}$ perquè la classe dels kernels és tancada pel producte tensorial. Ara, per ser R finit aleshores, pel Lema 12,

$$\sum_{\hat{g} \in R^{-1}(G)} \sum_{\hat{g'} \in R^{-1}(G')} \hat{K}(\hat{g},\hat{g'})$$

és un kernel a $\mathcal{P}(\{R^{-1}(g) : g \in \mathcal{G}\}) \times \mathcal{P}(\{R^{-1}(g) : g \in \mathcal{G}\})$. Notem que si consideram la seva zero extensió aleshores és un kernel a $\mathcal{G} \times \mathcal{G}$ com volíem veure. □

El R-convolucional kernel es pot expressar en termes de la descomposició del graf en conjunts de subestructures S de la forma següent:

$$k_R(G,G') = \sum_{s \in S(G)} \sum_{s' \in S'(G')} \prod_{i=1}^{d} K_i(s,s')$$

on S i S' denoten subestructures de G i G' (respectivament) i d és el nombre de conjunts de subestructures diferents que considerarem. Aquestes subestructures poden ser el conjunt de nodes, el conjunt d'arestes, el conjunt format pel camí més curt entre cada parell de nodes del graf, etc.

Els R-convolucional kernels són de gran utilitat per a comparar grafs a partir de les seves subestructures. Vegem-ne un exemple que és el **Subtree Kernel**, un kernel bàsic basat en la comparació d'arbres.

Definició 21 *Sigui $\mathcal{T}$ un conjunt finit d'arbres (no necessàriament del mateix ordre), donats dos arbres $T, T' \in \mathcal{T}$ el kernel $K_{subtree} : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ es defineix com*

$$K_{subtree}(T,T') = \sum_{S \in Desc(T)} \sum_{S' \in Desc(T')} K_{tree}(S,S')$$

on $Desc(T)$ és el conjunt de tots els subarbres de T (Definició 42) i $K_{tree}(S,S')$ val 1 si S i S' són iguals i 0 en cas contrari.

El Subtree Kernel de dos arbres ens dona el nombre de subarbres que tenen en comú els arbres.

Proposició 22 *El Subtree Kernel és un kernel de grafs.*

Prova.

Observem que $K_{subtree}$ és un kernel per ser un R-Convolutional kernel on S i S' contenen un sol conjunt de subestructures ($d = 1$) que és el conjunt de la descomposició en arbres de T i T' (respectivament) i $K_1 = K_{tree}$. □

Per acabar aquesta secció, a la Taula 2.1 mostram els diferents kernels per grafs que existeixen, el seu cost computacional i les característiques dels grafs als quals es poden aplicar [2, 3].

Com que l'objectiu d'aquest treball és comparar m-DAGs, que són grafs dirigits acíclics amb etiquetes als nodes, considerarem només alguns dels kernels de la taula. En particular, considerarem el Node Histogram, Shortest-Path, Weisfeiler-Lehman, ODD-STh i Pyramid Match kernel.

Finalment, ja que l'objectiu d'aquest treball és analitzare el comportament de diferents kernels aplicats als m-DAGs, per tal de poder comparar els resultats obtinguts per cada kernel, cal normalitzar la matriu de kernels.

Kernel	dirigit		nodes		arestes		ordre computacional
	si	no	etiquetes	atributs	etiquetes	atributs	
All node-pairs	•	•	•	•			$O(n^2 d_v)$
Node Histogram	•	•	•	•			$O(nd_v)$
All edge-pairs	•	•			•	•	$O(m^2 d_e)$
Edge histogram	•	•			•	•	$O(m^2 d_e)$
Shortest-path	•	•	•	•			$O(n^4 d_v)$
GraphHopper	•	•	•	•			$O(n^4)$
Subtree pattern	•	•	•				$O(n^2 h 4^d)$
Cyclic pattern	•	•	•		•		$O((c+2)n+2m)$
Graph edit distance	•	•	•	•	•	•	$O(n^3)$
Graphlet	•	•					$O(nd^{k-1})$
Direct product graph	•	•	•		•		$O(n^6)$
Marginalized random walk	•	•	•		•		$O(n^6)$
Random walk	•	•	•		•	•	$O(n^3)$
Quantum walk	•	•					$O(n^3)$
Weisfeiler–Lehman	•	•	•		•		$O(hm)$
Neighbourhood hash	•	•	•		•		$O(hm)$
Neighbourhood subgraph pairwise distance		•	•		•		$O(nn_h m_h \log(m_h))$
Hadamerd code	•	•	•		•		$O(hm)$
Propagation framework	•	•	•	•	•	•	$O(hm)$
Message passing	•	•	•	•			$O(n^2)$
Multiscale Laplacian		•	•			•	$O(n^2 h)$
Subgraph matching	•	•	•	•	•	•	$O(k(n^2)^{k+1})$
Graph invariant framework	•	•	•	•	•	•	$O(\tau n^2 d^{4r})$
Hash graph kernels	•	•	•	•	•	•	-
Weighted decomposition	•	•	•	•	•	•	$O(l^2)$
Optimal assignment	•	•	•		•		$O(hm)$
Deep graph kernels	•	•	•	•	•	•	-
Core based kernel framework		•	•	•	•	•	-
Pyramid match	•	•	•	•	•	•	$O(d_v m \log(2^h d))$

Taula 2.1: La taula ens classifica els kernels en funció de si els grafs són dirigits o no, si tenen etiquetes o atributs als nodes o arestes i el seu cost computacional amb la notació Big O que és la cota superior de la complexitat del temps d'execució de l'algoritme en funció dels valors d'entrada. Cal dir que definim n com l'ordre del graf, m la mida, d el grau màxim, d_v i d_e dimensió de les etiquetes dels nodes i de les arestes respectivament, k tamany del subgraf, c fita superior del nombre de cicles de qualsevol graf, r diàmetre, τ nombre màxim de subestructures, l nombre màxim de selector-context i h nombre d'iteracions.

Definició 23 (Matriu de Kernels Normalizada) *Definim la matriu normalitzada de K com $\hat{K}$ definida per*

$$\hat{K}_{ij} = \frac{K(x_i, x_j)}{\sqrt{K(x_i, x_i)}\sqrt{K(x_j, x_j)}}.$$

Proposició 24 *La matriu de Kernels Normalizada és una matriu de kernels.*

Podeu consultar la prova de que $\hat{K}$ satisfà les condicions necessàries per ser un kernel [6].

CAPÍTOL 3

KERNELS PER M-DAGS

En aquest capítol introduirem els kernels que hem escollit per comparar els m-DAGs. Els kernels escollits són el Node Histogram Kernel (NH), el Shortest-Path Kernel (SP), el Weisfeiler-Lehman Kernel (WL), el Ordered Decomposed DAGs kernel (ODD-Sth) i el Pyramid Match Kernel (PM), tots aquests kernels es poden aplicar per comparar grafs dirigits i etiquetats. Hem considerat el Node Histogram Kernel perquè és el kernel més senzill per grafs etiquetats, el Shortest-Path Kernel perquè, de manera gradual afegeix informació estructural a la comparació dels grafs, però, conceptualment també és molt senzill. Seguidament, hem considerat el Weisfeiler-Lehman Kernel perquè és un dels kernels més populars i més ben considerats. Hem afegit el Ordered Decomposed DAGs kernel perquè considera la descomposició primer en DAGs i després en arbres, dels grafs a comparar, el que ens sembla adient, a priori, ja que nosaltres volem comparar DAGs. Finalment, hem considerat també el Pyramid Match Kernel perquè, a diferència dels altres kernels, aquest considera propietats espectrals dels grafs a comparar.

A continuació definirem formalment cada un dels kernels escollits i en donarem exemples molt senzills per facilitar-ne la comprensió. Tot i que els kernels els aplicarem a DAGs, aquests kernels estan definits tant per grafs dirigits com per grafs no dirigits, per tant, els exemples els farem amb grafs no dirigits, perquè és la versió més general.

3.1 Node Histogram Kernel

El *Node Histogram Kernel*, un dels kernels més senzills definit per a grafs etiquetats, únicament considera les etiquetes dels nodes del graf com a vector associat al graf [2].

Definició 25 (Node Histogram Kernel) *Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt de les etiquetes. Per a cada $G = (V, E) \in \mathcal{G}$, si $l : V \to L$ és la funció que assigna una etiqueta a cada node del graf, es defineix* l'histograma d'etiquetes *del graf G, com el vector $\phi(G) = (c(1), c(2), ..., c(d)) \in \mathbb{R}^d$ tal que*

$$c(i) = |\{v \in V : l(v) = \theta_i\}|.$$

Siguin $\phi(G)$ i $\phi(G')$ els histogrames d'etiquetes dels grafs G i G', respectivament, llavors el Node Histogram Kernel està definit com

$$k_{NH}(G,G') = \langle \phi(G), \phi(G') \rangle$$

on $\langle , \rangle$ denota el producte escalar habitual a $\mathbb{R}^d$.

Proposició 26 *El Node Histogram Kernel és un kernel de grafs.*

Prova. Donat un conjunt finit de grafs etiquetats, $\mathcal{G}$, el seu conjunt d'etiquetes L és un conjunt finit i el seu cardinal, d, queda fixat. Llavors, podem considerar l'aplicació $H : \mathcal{G} \to \mathbb{R}^d$ que a cada graf li assigna el seu histograma d'etiquetes, que és un vector de longitud d, i per tant per les propietats del producte escalar, clarament $k_{NH}(G,G') = \langle \phi(G), \phi(G') \rangle$ és un kernel per la Proposició 5. □

Observem que, si els grafs G i G' estan injectivament etiquetats, és a dir, nodes diferents tenen etiquetes diferents, llavors cada etiqueta només apareix un cop i, per tant, els vectors $\phi(G)$ i $\phi(G')$ són vectors binaris pel que $k(G,G') = \langle \phi(G), \phi(G') \rangle$ ens dona el nombre d'etiquetes que tenen en comú els dos grafs.

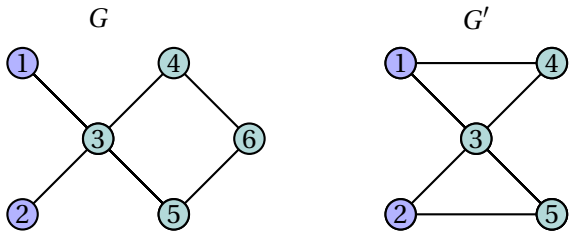
Proposició 27 *Donats dos grafs etiquetats $G, G' \in \mathcal{G}$, sigui L el seu conjunt d'etiquetes L, llavors el cost computacional del Node Histogram Kernel és $O(nd)$, on $d = |L|$ i $n = max(|V(G)|, |V(G')|)$.*

Prova. Simplement cal observar que per obtenir $\phi(G)$, l'algoritme ha de recórrer el conjunt d'etiquetes de cardinal d per cada node de G .

□

Vegem ara un exemple senzill de l'aplicació del Node Histogram Kernel.

Exemple 28 *Siguin G i G' els grafs no dirigits i etiquetats següents on els colors dels nodes denoten l'etiqueta, mentre que l'ordenació dels nodes és arbitrària.*



els grafs estan etiquetats amb només dues etiquetes A i B, per la qual cosa $d = 2$ i l'histograma d'etiquetes corresponent és el que mostra a la taula següent:

$(h = 0)$	●	●	H
Etiqueta	A	B	
Nº de nodes de G	2	4	(2,4)
Nº de nodes de G'	2	3	(2,3)

Per tant, $K_{NH}(G,G') = \langle (2,4), (2,3) \rangle = 4 + 12 = 18$.

3.1.1 Shortest-path Kernel

El shortest path kernel considera la connectivitat entre parelles de nodes dels grafs a comparar i les seves etiquetes. És un kernel més elaborat que el node histogram, que únicament considera les etiquetes dels nodes. Donat un conjunt finit de grafs, per a cada graf es defineix el seu *shortest-path graph* que és un graf etiquetat amb pesos a les arestes. Llavors, es defineix el shortest-path kernel de dos grafs comparant els seus shortest-path graphs.

Definició 29 (Shortest-path Graph) *Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt de les etiquetes. Per a cada $G = (V, E) \in \mathcal{G}$, si $l : V \to L$ és la funció que assigna una etiqueta a cada node del graf, definim el* shortest-path graph *de G, $S_G = (V, E_S)$, com el graf que té els mateixos nodes amb les mateixes etiquetes que G i, $uv \in E_S$ si, i només si, existeix un camí que connecta u i v a G. Definim el pes de l'aresta $uv \in E_S$ com la distància de u a v dins G. Denotam per $\mathcal{G}_s$ el conjunt de shortest-path graph de G.*

A continuació donam la definició del Shortest-Path kernel a partir de la definició donada a [12].

Definició 30 (Shortest-path Kernel) *Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt de les etiquetes. Donats dos grafs $G = (V, E), G' = (V', E') \in \mathcal{G}$, siguin $S_G = (V, E_S)$ i $S_{G'} = (V', E'_S)$ els seus respectius shortest-path graphs. Aleshores, el shortest-path kernel de G i G' es defineix com*

$$K_{SP}(G, G') = \sum_{e \in E_S} \sum_{e' \in E'_S} k_{path}(e, e').$$

És a dir, el shortest-path kernel compara la distància entre parells de nodes amb les mateixes etiquetes a G i G'.

Proposició 31 *El Shortest-path kernel és un kernel de grafs.*

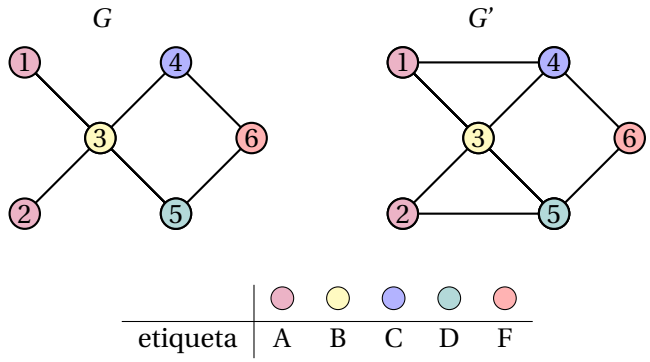
Prova. Per la Proposició 17, sabem que K_{path} és un kernel en el conjunt d'arestes de $\mathcal{G}_s$. Llavors, per la Definició 18, k_{SP} és un kernel per ser un R-Convolutional kernel considerant un sol conjunt de subestructures ($d = 1$), que és el conjunt de les arestes (juntament amb els seus pesos) dels Shortest-path graphs de G i G' (respectivament), i $K_1 = K_{path}$. □

Proposició 32 *Donats dos grafs $G, G' \in \mathcal{G}$, el cost computacionl del Shortest-path Kernel és $O(n^4)$, on $n = max(|V(G)|, |V(G')|)$.*

Prova. Observem que el cost computacional d'aquest kernel depèn del nombre d'arestes dels grafs G i G'. En el pitjor dels casos, si existeix una aresta entre cada parella de nodes, per calcular el Shortest-path graph s'hauran de recórrer totes i l'algoritme tendrà ordre quadràtic (Algorisme de Dijkstra [13]). Aleshores, per calcular el kernel tenim dos sumatoris que recorren totes les arestes dels Shortest-path graphs. Llavors, en el pitjor dels casos, si $m = n^2$, el cost de calcular el kernel és $O(n^4)$, on n és la mida del graf de major nombre de nodes [12]. □

Vegem ara un exemple senzill de l'aplicació del Shortest-path Kernel.

Exemple 33 *Siguin G i G' els grafs no dirigits i etiquetats de sota on els colors dels nodes denoten l'etiqueta, mentre que l'ordenació dels nodes és arbitrària.*



Aleshores, els grafs ponderats S_G i $S_{G'}$ tenen les següents matrius d'adjacència A i A' respectivament, amb pesos a les arestes que es corresponen a les distàncies entre els nodes.

$$A = \begin{pmatrix} 0 & 2 & 1 & 2 & 2 & 3 \\ 2 & 0 & 1 & 2 & 2 & 3 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 2 & 2 & 1 & 0 & 2 & 1 \\ 2 & 2 & 1 & 2 & 0 & 1 \\ 3 & 3 & 2 & 1 & 1 & 0 \end{pmatrix} \qquad A' = \begin{pmatrix} 0 & 2 & 1 & 1 & 2 & 2 \\ 2 & 0 & 1 & 2 & 1 & 2 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 1 & 2 & 1 & 0 & 2 & 1 \\ 2 & 1 & 1 & 2 & 0 & 1 \\ 2 & 2 & 2 & 1 & 1 & 0 \end{pmatrix}$$

Notem que cada element a_{ij} de les matrius indica el pes de l'aresta entre els nodes v_i i v_j, és a dir, la distància entre els nodes v_i i v_j a G si aquests estan connectats i zero si no hi estan. A més, observem també que $k_{node}(v_i, v'_k) = 1$ si, i només si, $i = k$. Per tant, tenim que $k_{path}(e, e') = 1$ quan $e = v_i u_j$, $e' = v'_i u'_j$ i tenen el mateix pes.

Així doncs,

$$k_{SP}(G,G') = \sum_{i,j\in\{1,\dots,6\}} \sum_{k,m\in\{1,\dots,6\}} k_{node}(v_i,v'_k)\cdot k_{edge}(e_{ij},e'_{km})\cdot k_{node}(u_j,u'_m) =$$

$$\sum_{j,m\in\{1,\dots,6\}} k_{node}(v_1,v'_1)\cdot k_{edge}(e_{1j},e'_{1m})\cdot k_{node}(v_j,v'_m)+\cdots+$$

$$\sum_{j,m\in\{1,\dots,6\}} k_{node}(v_6,v'_6)\cdot k_{edge}(e_{6j},e'_{6m})\cdot k_{node}(v_j,v'_m)+$$

$$\sum_{j,m\in\{1,\dots,6\}} k_{node}(v_1,v'_2)\cdot k_{edge}(e_{1j},e'_{2m})\cdot k_{node}(v_j,v'_m)+$$

$$\sum_{j,m\in\{1,\dots,6\}} k_{node}(v_2,v'_1)\cdot k_{edge}(e_{2j},e'_{1m})\cdot k_{node}(v_j,v'_m)$$

perquè tots els altres sumatoris s'anul·len.

Finalment, podem observar que si comparant els valors de les matrius a la triangular superior, per files obtenim que els valors de coincidència de pesos és:

$$k_{SP}(G,G') = 3+3+7+5+5+3+3+3 = 32$$

3.1.2 Weisfeiler-Lehman Kernel

El Weisfeiler-Lehman Kernel està basat en el Weisfeiler-Lehman test, que és un test dissenyat per poder decidir si dos grafs etiquetats són isomorfs a partir d'un reetiquetatge astut de les etiquetes originals dels grafs [2, 14].

Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt de les etiquetes. Per a cada $G = (V,E) \in \mathcal{G}$, $l : V \to L$ és la funció que assigna una etiqueta a cada node del graf. El test de Weisfeiler-Lehman d'isomorfia de grafs consisteix en augmentar iterativament les etiquetes dels nodes dels grafs de manera que l'etiqueta augmentada és una tupla formada per l'etiqueta del node original i el multiconjunt de les etiquetes dels veïnats d'aquest node. A cada pas d'iteració, la nova etiqueta es comprimeix, i el procés es repeteix fins que els multiconjunts d'etiquetes dels dos grafs són diferents (indicant que els grafs no poden ser isomorfs), o fins que s'hagi assolit el nombre màxim d'iteracions $h_{max} := \max\left(\{|V(G)| : G \in \mathcal{G}\}\right)$. Aquest procediment garanteix produir seqüències idèntiques per a grafs isomorfs. Cal observar però, que en general és possible que dos grafs no isomorfs també tinguin seqüències idèntiques, tal com mostram a l'Exemple 39.

Definició 34 *Sigui $G \in \mathcal{G}$. Definim, de manera iterativa, un reetiquetatge dels nodes del graf G de la manera següent:*

- *El cas base ($h = 0$) d'aquesta iteració utilitza les etiquetes originals del graf, de manera que $l(v)^{(0)}_{WL} := l(v)$.*
- *Per a $h > 0$, a cada node se li assigna una nova etiqueta que identifica de manera única la tupla formada per l'etiqueta actual de Weisfeiler-Lehman del node, $l(v)^{(h)}_{WL}$, i el multiconjunt d'etiquetes actuals de Weisfeiler-Lehman dels seus veïnats, $\left\{l(v_0)^{(h)}_{WL} | v_0 \in N(v)\right\}$, és a dir, les actualitzacions tenen la forma*

$$l(v)^{(h+1)}_{WL} := f\left(l(v)^{(h)}_{WL}, \left\{l(v_0)^{(h)}_{WL} | v_0 \in N(v)\right\}\right),$$

  on f és una funció que comprimeix la tupla en una única etiqueta amb valor enter.

Aleshores, l'operació de reetiquetatge de Weisfeiler-Lehman de G resulta en una seqüència de grafs

$$G_0 \to G_1 \to \cdots \to G_h$$

coneguda com la seqüència de Weisfeiler-Lehman on cada graf només difereix dels altres en termes de les seves etiquetes.

A l'exemple 38 mostram com s'obté aquesta seqüència de grafs. A continuació, donam la definició del Weisfeiler-Lehman kernel en la seqüència de grafs construida.

Definició 35 (Weisfeiler-Lehman kernel) *Siguin $G, G' \in \mathcal{G}$ i sigui $h \in \mathbb{N}$, definim el kernel de Weisfeiler-Lehman com*

$$k_{WL}^{(h)}(G, G') := \sum_{i=0}^{h} k(G_i, G_i')$$

on G_i i G_i' són el i-èssim graf de les seqüències de Weisfeiler-Lehman de G i G', respectivament, i k és un kernel base definit per a grafs.

Proposició 36 *El Weisfeiler-Lehman kernel és un kernel de grafs.*

Prova. Donat el conjunt finit de grafs etiquetats $\mathcal{G}$, una vegada calculada la seqüència de Weisfeiler-Lehman dels seus grafs, considerem $L_{WL}^{(h)}(G) := \{\theta_1^{(h)}, \theta_2^{(h)}, \ldots\}$ l'alfabet de totes les etiquetes comprimides en el pas h de l'operació de reetiquetatge de Weisfeiler-Lehman de G i, $L_{WL}(G) = \bigcup_{h=1}^{d} L_{WL}^{(h)}$ el conjunt de totes les etiquetes de la seqüència de Weisfeiler-Lehman del graf G. Sigui $L_{WL}(\mathcal{G})$ la unió dels conjunts d'etiquetes de Weisfeiler-Lehman de tots els grafs de $\mathcal{G}$, i sigui $\mathcal{G}_{WL}$ el conjunt format per totes les seqüències de WL dels grafs de $\mathcal{G}$ (cal recordar que els grafs d'una mateixa seqüència només difereixen en les etiquetes). Llavors, el kernel base del kernel de WL és un kernel a $\mathcal{G}_{WL} \times \mathcal{G}_{WL}$ i per tant, per la Proposició 13, K_{WL} és un kernel de grafs per ser suma de kernels de grafs a $\mathcal{G}_{WL} \times \mathcal{G}_{WL}$. □

A partir d'aquesta definició podem obtenir multitud de kernels diferents. No obstant això, el més comú és el que usa com a kernel base el Node Histogram Kernel. Sigui $l : V \to L_{WL}(\mathcal{G})$ la funció d'etiquetatge d'un graf donat G, el Node Histogram Kernel utilitza una funció $c(i) = |\{v \in V : l(v) = \theta_i\}|$ per comptar el nombre d'aparicions d'una etiqueta θ_i de $L_{WL}(\mathcal{G})$. Aleshores definim el Kernel de Weisfeier Lehman amb kernel base el Node Histogram kernel com

$$k_{WL_{NH}}(G, G') := \langle \phi(G), \phi(G') \rangle$$

on hem definit el vector de característiques de les seqüències de WL com $\phi(G) = (c(1), c(2), ..., c(D)) \in \mathbb{R}^D$ on D és el cardinal de $L_{WL}(\mathcal{G})$.

Proposició 37 *Donats dos grafs $G, G' \in \mathcal{G}$, el cost computacional de h iteracions de reetiquetatge de Weisfeiler-Lehman és $O(hm)$ on $m = max(|E(V)|, |E(V')|)$.*

Prova. En cada iteració, el cost de reetiquetatge de Weisfeiler-Lehman és $O(m)$ on m és la mida del graf ja que es recorren totes les arestes. Així doncs, el cost de h iteracions del WL Kernel, sense comptar cap càlcul de kernel, és $O(hm)$ [14].

□

Cal observar que, el càlcul del Kernel de Weisfeiler-Lehman també depèn del kernel base seleccionat. Si consideram el Node Histogram com a kernel base, llavors el cost és també $O(hm)$ ja que el cost del Node Histogram Kernel és més baix i el Big O es queda amb els termes de major grau, recordem que el cost del Node Histogram Kernel és $O(dn)$ on d és el cardinal del conjunt d'etiquetes i n l'ordre màxim dels grafs.

Per tal d'entendre millor aquest kernel, vegem-ne un exemple.

Exemple 38 *Siguin G i G' els grafs no dirigits i etiquetats de la Figura 3.1 on els colors dels nodes denoten l'etiqueta, mentre que l'ordenació dels nodes és arbitrària.*

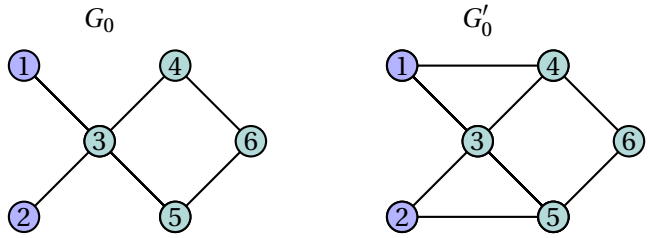


Figura 3.1: Grafs del la iteració $h = 0$ de WL

A l'inici (h=0) els nodes tenen l'etiqueta A o l'etiqueta B.

$(h = 0)$		
Etiqueta	A	B
Nº de nodes de G_0	2	4
Nº de nodes de G'_0	2	4

Al pas $h = 1$ afegim a cada etiqueta del node les etiquetes dels seus veïnats, tenim doncs el grafs etiquetats de la Figura 3.2 amb les cinc noves etiquetes:

$(h = 1)$					
Etiqueta	A,B	B,AABB	B,BB	A,BB	B,BBA
Nova etiqueta	C	D	E	F	G
Nº nodes de G_1	2	1	3	0	0
Nº nodes de G'_1	0	1	1	2	2

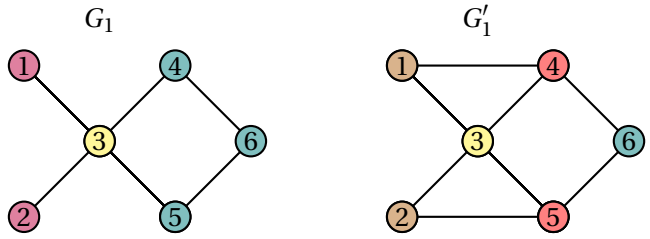


Figura 3.2: Grafs de la iteració $h = 1$ de WL

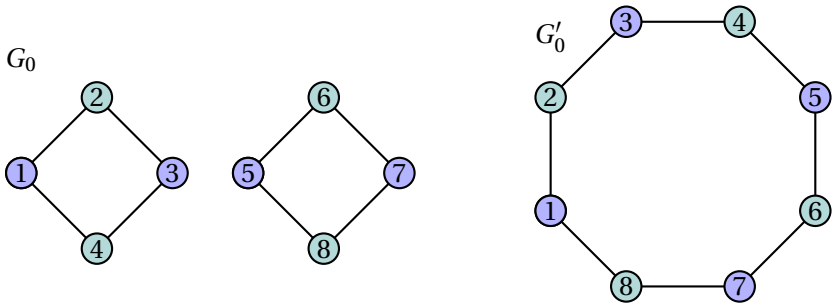
I, per tant tenim que

$$k_{WL}^{(1)}(G,G') = \langle \phi(G), \phi(G') \rangle = \langle (2,4,2,1,3,0,0), (2,4,0,1,1,2,2) \rangle = 24.$$

Repetint aquest procés, en cada iteració obtenim un reetiquetatge G_i i G'_i de G i G' respectivament, i per tant un valor de $K(G_i, G'_i)$.

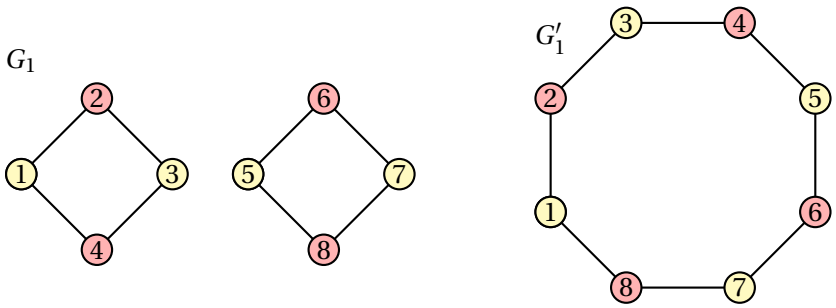
Per acabar, vegem un exemple on grafs no isomorfs tenen les mateixes etiquetes en les seqüències de Weisfeiler-Lehman.

Exemple 39 *Considerem els grafs següents no dirigits amb etiquetes A o B als seus nodes i que no són isomorfs, ja que G_0 no és connex i G' sí que ho és.*



(h=0)	●	●
Etiqueta	A	B
Nº nodes de G_1	4	4
Nº nodes de G'_1	4	4

Calculam la següent iteració del mètode de Weisfeisler Lehman, que consisteix en afegir a cada node les etiquetes dels seus veïnats. Llavors tenim els grafs etiquetats següents.



(h=1)	●	●
Etiqueta	B,AA	A,BB
Nova Etiqueta	C	D
Nº nodes de G_1	4	4
Nº nodes de G'_1	4	4

Notem que tornam a tenir la mateixa situació del principi i multiconjunts d'etiquetes idèntics. És a dir, els grafs etiquetats G_0 i G_1 són isomorfs i els grafs etiquetats G'_0 i G'_1 també ho són, i per tant $K(G_i, G'_i) = K(G_0, G'_0)$ per a tot $i \in \{1, ..., h\}$.

3.1.3 ODD-STh Kernel

La idea del *Ordered Directed DAG Kernel* (ODD-STh Kernel) és fer ús dels kernels definits en DAGs i arbres considerant aquests com a subestructures dels grafs a comparar. Per això, cal primer introduir el concepte de descomposició d'un graf en DAGs ordenats.

Definició 40 (Descomposició en DAGs d'un graf) *Donat un graf $G = (V, E)$ d'ordre n, aquest es descompon en n DAGs, un per a cada node de G, de la manera següent: el DAG i-èssim de la descomposició corresponent al node v_i de G, D_{v_i}, està format pels nodes de G tals que existeix un camí des de v_i fins aquests i per les arestes que pertanyen als camins dirigits més curts des de v_i a tots els altres nodes del graf G. Anomenam $DAGs(G)$ al conjunt que emmagatzema els DAGs de la descomposició de G.*

Cal recalcar que si no existeix un camí des d'un node u fins a un altre node v d'un graf, ni v estarà al DAG corresponent a u a la descomposició en DAGs de G ni al revés.

Exemple 41 *A la Figura 3.3 es mostren les descomposicions en DAGs del graf línia i d'un graf cicle, ambdós de 4 nodes. Per cada graf obtenim 4 DAGs, un per cada node.*

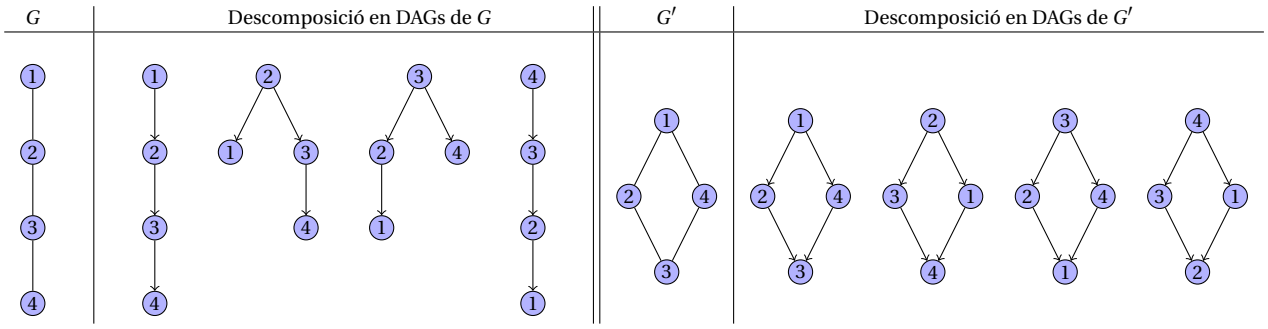


Figura 3.3: Descomposicions en DAGs del graf línia i d'un cicle d'ordre 4.

A continuació introduim la descomposició d'un DAG d'ordre n en n arbres.

Definició 42 (Descomposició en arbres d'un DAG) *Donat un DAG D d'orde n, l'arbre i-èssim de la descomposició de D, $T_D(v_i)$, és un arbre arrelat a v_i, on els nodes són els nodes u de D tals que existeix un camí a D de v_i a u, i les arestes d'aquest camí pertanyen a l'arbre.*

Exemple 43 *A la Figura 3.4 es mostra la descomposició en arbres dels DAGs de la Figura 3.3. Podem observar que tenim un arbre per a cada node de cada DAG i, pels node fulla, l'arbre corresponent és només el node fulla. També veim que a D', quan hi ha un node fulla amb grau d'entrada 2, a l'arbre corresponent a l'arrel, la fulla es repeteix 2 vegades.*

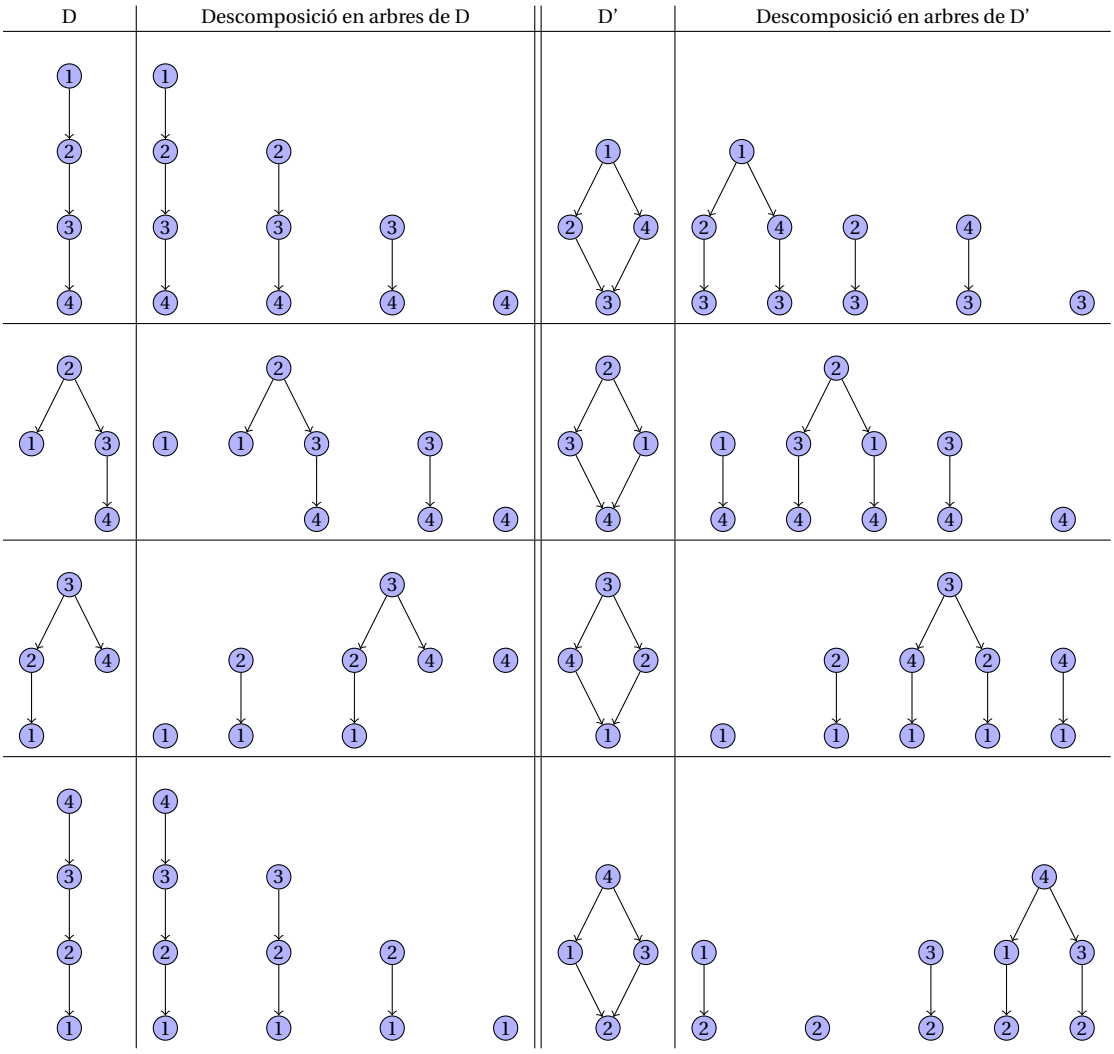


Figura 3.4: A la figura es mostren les descomposicions en arbres dels DAGs de la descomposició en DAGs de la Figura 3.3.

Definició 44 (ODD-STh Kernel) *Sigui $\mathcal{G}$ un conjunt finit de grafs etiquetats i sigui $L = \{\theta_1, \theta_2, ..., \theta_d\}$ el conjunt d'etiquetes. Donats dos grafs $G, G' \in \mathcal{G}$, el ODD-STh Kernel es defineix com*

$$K_{ODD-sth}(G, G') = \sum_{D \in DAGs(G)} \sum_{D' \in DAGs(G')} K_{DAG}(D, D')$$

on

$$K_{DAG}(D, D') = \sum_{v \in V(D)} \sum_{v' \in V(D')} K_{subtree}(T_D(v), T_{D'}(v')).$$

Proposició 45 *El ODD-sth kernel és un kernel de grafs.*

Prova. Observem primer que, donats dos DAGs D i D', $K_{DAG}(D, D')$ és un R-Convolutional kernel, ja que per la Definició 18, considerant un sol conjunt de subestructures ($d = 1$), llavors S i S' són els arbres de la descomposició en arbres de D i D' de la Definició 42 i $K_1 = K_{subtree}$. Llavors, $K_{ODD-sth}(G, G')$ és també un R-Convolutional kernel

considerant S i S' un sol conjunt de subestructures que és el conjunt de DAGs de la descomposició en DAGs de la Definició 40 i $K_1 = K_{DAG}$. □

Notem que l'ordre computacional d'aquest kernel depen de la definició de $K_{subtree}$ que empleam. En el nostre cas, usam la donada a la Definició 21.

Proposició 46 *Donats dos grafs $G, G' \in \mathcal{G}$, l'ordre computacional de la primera versió del ODD-sth kernel és $O(n^4)$ on $n = max(|V(G)|, |V(G')|)$ [15].*

Notem que l'algorisme que hem definit és bastant costòs. Per millorar l'eficiència es va proposar descompondre un graf G en un sol DAG al qual anomenam BigDAG [16]. El $BigDAG$ d'un graf G és el DAG que s'obté calculant el $BigDAG$ dels $DAGs$ de la descomposició de G. Donats dos DAGs $D, D' \in DAGs(G)$, el $BigDAG(D, D')$ és un DAG que conté tots els subarbres de D i D' només una vegada, és a dir, donat un arbre T comú als dos arbres aquest només apareix una vegada al DAG. Juntament amb el càlcul del BigDAG, calcularem també la freqüència d'aparició de cada subarbre del graf G. Definirem freqüència d'aparició del subarbre determinat per u a G, $f(u)$, com el nombre de descomposicions en arbres dels DAGs de G a les que pertany.

Exemple 47 *Vegem un exemple del càlcul del BigDAG, a la Figura 3.5 i Figura 3.6 podem observar com calcular el BigDAG dels grafs de l'Exemple 41.*

Notem que per calcular $BigDAG(G)$, calculam els $BigDAGs$ dels dos primers DAGs de la descomposició de G amb les respectives freqüències d'aparició dels subarbres (inicialment valen 1), i seguidament calculam el $BigDAG$ entre aquest i el tercer DAG i així amb tots els $DAGs$ de la descomposició de G fins a obtenir $BigDAG(G)$. Quan tenim un arbre que és comú a les descomposicions en DAGs dels dos DAGs, la seva freqüència al BigDAG resultant és la suma de les freqüències d'aquest a cada DAG.

Calculam $BigDAG(G')$ de forma anàloga. Notem que qualsevol parella de DAGs en la descomposició de G' no tenen cap arbre en comú (mirar Figura 3.4) i, per tant, les components connexes dels BigDAGs resultants són els mateixos DAGs.

Finalment, podem reescriure el ODD-sth Kernel com

$$K_{ODD-sth}(G, G') = \sum_{u \in BigDAG(G)} \sum_{u' \in BigDAG(G')} f(u) f'(u') K_{subtree}(T_{BigDAG(G)}(u), T_{BigDAG(G')}(u'))$$

on $f(u)$ i $f'(u')$ denoten la freqüència d'aparició dels subarbres $T_{BigDAG(G)}(u)$ i $T_{BigDAG(G')}(u')$ a $BigDAG(G)$ i $BigDAG(G')$, respectivament.

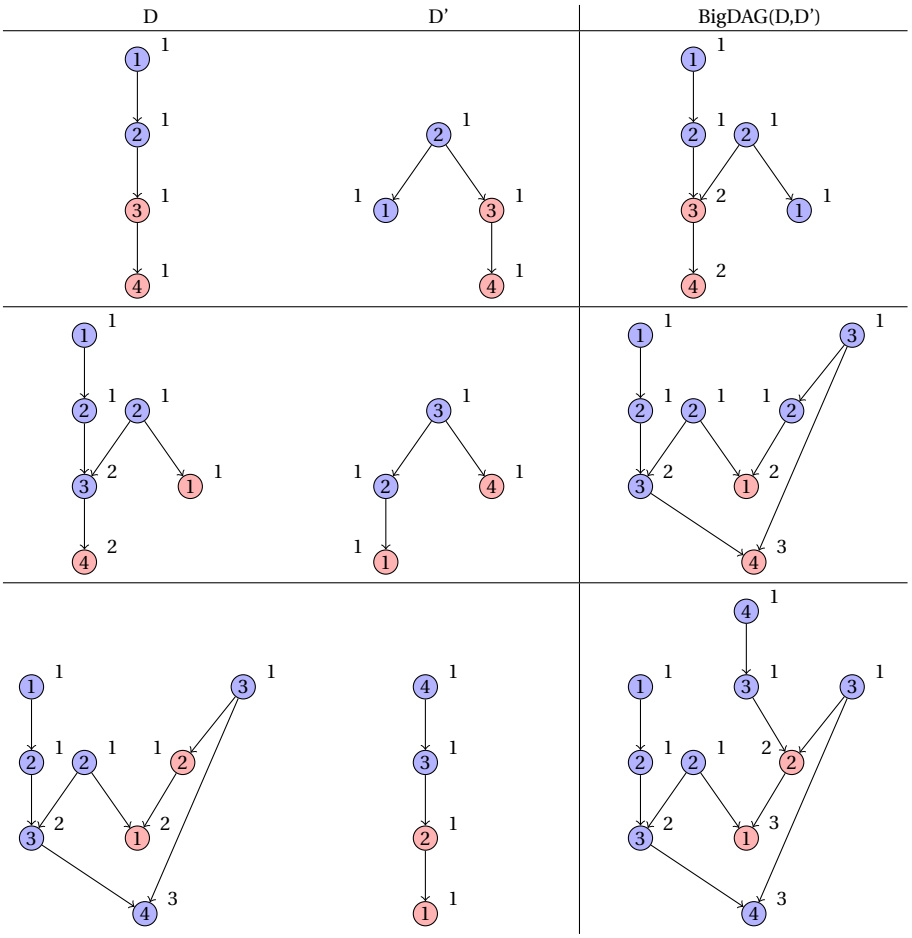


Figura 3.5: Podem veure un exemple del càlcul del $BigDAG(G)$. Notem que donats dos DAGs, es mira si les seves descomposicions en arbres (Figura 3.4) tenen elements comuns (marcats en vermell) i aquests apareixen una sola vegada al BigDAG. Els nombres de devora els nodes emmagatzemen la freqüència.

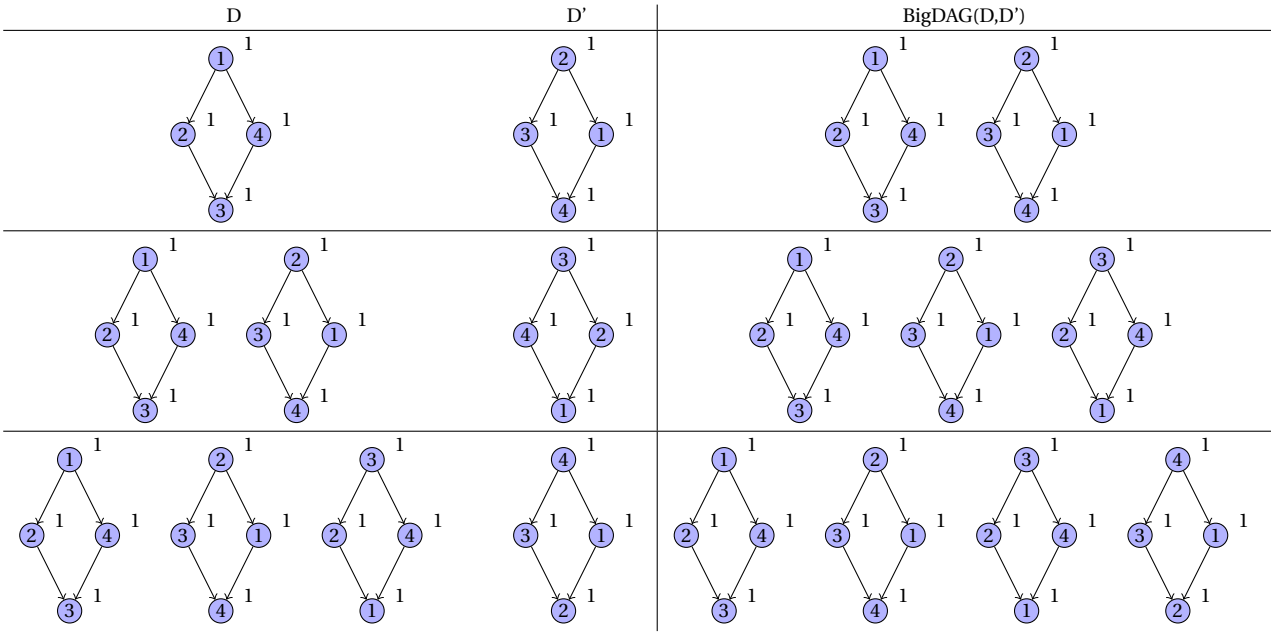


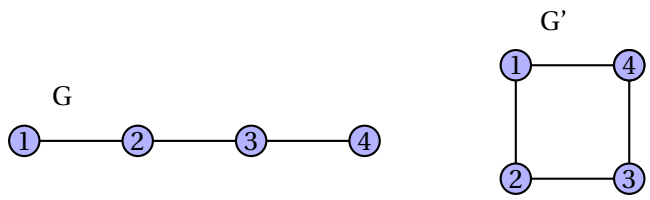
Figura 3.6: Podem veure un exemple del càlcul del $BigDAG(G')$.

3.1.4 Pyramid Match Kernel

La idea del Pyramid Match Kernel consisteix en interpretar els nodes dels grafs com a punts d'un espai euclidià, a partir dels valors i vectors propis de les seves matrius d'adjacència, i, després, calcular el kernel a partir d'aquests punts [17, 18].

Primer, transformam els nodes del graf en vectors de dimensió més baixa usant els d vectors propis de magnitud major de la matriu d'adjacència.

Definició 48 (Conjunt de punts a espai euclidià d'un graf) *Donat un graf G, sigui A_G la seva matriu d'adjacència i sigui P_G la matriu que té per columnes els vectors propis de A_G normalitzats i ordenats en funció de l'ordre dels valors propis (de major a menor). Aleshores, definim el conjunt de punts $P(G) = \{u_1, \ldots, u_n\}$ com $u_i = (u_i^1, \ldots, u_i^d) \in \mathbb{R}^d$ amb u_i^j l'element j-èssim de la fila i-èssima de la matriu P_G en valor absolut.*



Exemple 49

Figura 3.7

Vegem un exemple de com calcular el conjunt de punts a $\mathbb{R}^d$, on $d = 2$, dels grafs de la Figura 3.7. Primer, calculam les seves matrius d'adjacència.

$$A_G = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad A_{G'} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Ara, calculam les matrius de vectors propis de G i G′ segons la Definició 48.

$$P_G = \begin{pmatrix} -0.371 & -0.601 & 0.601 & 0.371 \\ -0.601 & -0.371 & -0.371 & -0.601 \\ -0.601 & 0.371 & -0.371 & -0.601 \\ -0.371 & 0.601 & 0.601 & -0.371 \end{pmatrix} \qquad P_{G'} = \begin{pmatrix} -0.5 & 0 & 0 & -0.5 \\ -0.5 & 0.707 & -0.707 & 0.5 \\ -0.5 & 0 & 0 & -0.5 \\ -0.5 & -0.707 & 0.707 & 0.5 \end{pmatrix}$$

I, per tant, el conjunt de punts a $\mathbb{R}^d$ dels grafs són:

G	G'
$u_1 = (0.371, 0.601)$	$u'_1 = (0.5, 0)$
$u_2 = (0.601, 0.371)$	$u'_2 = (0.5, 0.707)$
$u_3 = (0.601, 0.371)$	$u'_3 = (0.5, 0)$
$u_4 = (0.371, 0.601)$	$u'_4 = (0.5, 0.707)$

Notem que els punts de la definició anterior es troben a l'hipercub unitari d-dimensional.

Definició 50 (Divisió en cel·les d'un hipercub unitari d-dimensional al nivell l) *Donat un hipercub unitari d-dimensional i donat l un nombre natural, dividirem cada dimensió de l'hipercub en dues parts iguals ($l = 1$), després, dividirem cada una d'aquestes parts en dues parts iguals ($l = 2$) i així de forma recursiva l vegades.*

Notem que al nivell l l'hipercub unitari d-dimensional té 2^l cel·les (de la mateixa grandària) a cada dimensió i té $D = 2^l d$ en cel·les en total. Ara, l'histograma emmagatzema quants de punts pertanyen a cada una de les cel·les en les quals dividim l'hipercub.

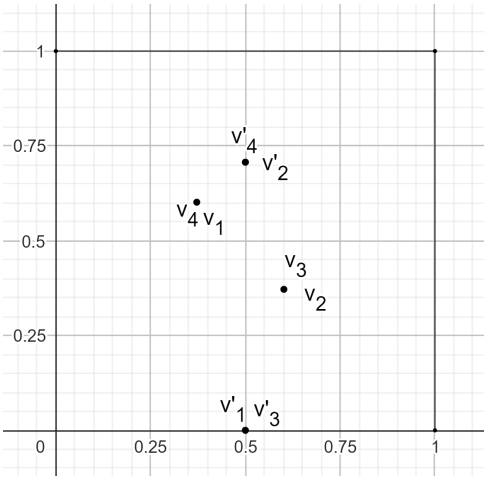
Definició 51 (Histograma d'un graf) *Donat un graf G, definim H_G^l com l'histograma de G al nivell l i $H_G^l(k)$ com el nombre de nodes que pertanyen a la cel·la $k \in \{1, \dots, D\}$ del nivell l. Formalment, si $k \in (2^l(i-1), 2^l i)$ per un cert $i \in \{1, \dots, d\}$, aleshores, $H_G^l(k)$ és el nombre de nodes tals que la seva coordenada i-èssima pertany a l'interval $\left[\frac{1}{2^l}(m-1), \frac{1}{2^l}m\right)$ on $m = k - 2^l i$.*

Definició 52 (Intersecció d'histogrames a un nivell) *Donats dos grafs G i G', definim*

$$I(H_G^l, H_{G'}^l) = \sum_{i=1}^{D} \min\left(H_G^l(i), H_{G'}^l(i)\right)$$

com el nombre de punts dels histogrames de G i G' que coincideixen al nivell l.

Exemple 53 *Vegem la representació del conjunt de punts del grafs de l'Exemple 49.*



A la taula següent podem veure els seus histogrames i intersecció d'aquests fins al nivell $l = 3$. El nivell $l = 0$ es correspon a tot el quadrat i, per tant, tots els punts hi pertanyen. Al nivell $l = 1$, hi ha dos punts, v_1 i v_4, de G tals que la primera coordenada és menor que $\frac{1}{2}$ mentre que a G' no n'hi ha cap i, per tant, tots cauen a la segona cel·la a la primera dimensió.

l	H_G^l	$H_{G'}^l$	$I(H_G^l, H_{G'}^l)$	$D = 2^l d$
0	$(4,4)$	$(4,4)$	8	2
1	$(2,2,2,2)$	$(0,4,2,2)$	6	4
2	$(0,2,2,0,0,2,2,0)$	$(0,0,4,0,2,0,2,0)$	4	8
3	$(0,0,2,0,2,0,0,0,0,0,2,0,2,0,0,0)$	$(0,0,0,0,4,0,0,0,2,0,0,0,0,2,0,0)$	2	16

A l'hora de definir el kernel, les coincidències a nivells més alts (cel·les més petites) tenen més pes a l'hora de calcular el kernel. El pes pel nivell l és $\frac{1}{2^{L-l}}$, és a dir, el pes del nivell és inversament proporcional a la grandària de les cel·les.

Definició 54 (Pyramid Match kernel) *Consideram $\mathcal{G}$ un conjunt finit de grafs no etiquetats i d, la dimensió del nou espai, i L, el nombre màxim de nivells, un nombre natural fixat. Aleshores, donats dos grafs $G, G' \in \mathcal{G}$, podem definir*

$$K_{Pyramid}(G,G') = I\left(H_G^L, H_{G'}^L\right) + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}} \left(I\left(H_G^l, H_{G'}^l\right) - I\left(H_G^{l+1}, H_{G'}^{l+1}\right)\right)$$

.

Exemple 55 *Calculem el kernel dels grafs de l'Exemple 49 per $d = 2$ i $L = 3$.*

$$K_{Pyramid}(G,G') = I\left(H_G^3, H_{G'}^3\right) + \frac{1}{2^3}\left(I\left(H_G^0, H_{G'}^0\right) - I\left(H_G^1, H_{G'}^1\right)\right) + \frac{1}{2^2}\left(I\left(H_G^1, H_{G'}^1\right) - I\left(H_G^2, H_{G'}^2\right)\right) +$$

$$\frac{1}{2}\left(I\left(H_G^2, H_{G'}^2\right) - I\left(H_G^3, H_{G'}^3\right)\right) = 2 + \frac{1}{2^3}(8-6) + \frac{1}{2^2}(6-4) + \frac{1}{2}(4-2) = 3.75.$$

En el cas de grafs etiquetats, si dos grafs coincideixen a la mateixa cel·la, només es té en compte si tenen la mateixa etiqueta. Donat un graf G i el seu conjunt de punts $P(G)$, consideram el conjunt format pels conjunts d'aquests punts agrupats en funció de la seva etiqueta ,$P_a(G)$. Aleshores, aplicam el pyramid match kernel als parells de conjunts associats a la mateixa etiqueta. Finalment, el kernel serà la suma d'aquests per totes les etiquetes, és a dir,

$$K_{Pyramid}(G,G') = \sum_{i=1}^{c} K^i_{Pyramid}(G,G')$$

on c és el nombre d'etiquetes i $K^i_{Pyramid}(G,G')$ és el pyramid match kernel considerant els conjunts de punts de $P_a(G)$ i $P_a(G')$ associats a l'etiqueta i, respectivament.

Proposició 56 *El Pyramid Match kernel és un kernel de grafs.*

Prova. Notem que podem reescriure el kernel de la forma

$$K_{Pyramid}(G,G') = \sum_{l=0}^{L-1} \left(\frac{1}{L-(l+1)} - \frac{1}{2^{L-l}} \right) I\left(H^l_G, H^l_{G'}\right) + \frac{1}{2^L} I\left(H^0_G, H^0_{G'}\right)$$

i, per tant, per la Proposició 13, es redueix a demostrar que la funció I és un kernel. Notem que I és simètric per ser suma de funcions simètriques i la prova de que I és semidefinida positiva consisteix en redefinir la funció I com el producte escalar de dos vectors. Notem que podem reescriure $I\left(H^l_G, H^l_{G'}\right) = \left\langle \mathcal{V}\left(H^l_G\right), \mathcal{V}\left(H^l_{G'}\right) \right\rangle$ on el vector $\mathcal{V}(H^l_G) \in \{0,1\}^{nD}$ considera n coordenades per cada coordenada $H^l_G(i)$ de l'histograma H^l_G, les $H^l_G(i)$ primeres coordenades valen 1 i les $n - H^l_G(i)$ darreres valen 0. Podeu consultar els detalls a [17]. □

Proposició 57 *Donats dos grafs $G, G' \in \mathcal{G}$, l'ordre computacional del Pyramid Match kernel és $O(dnL)$ on d és la dimensió del nou espai i L és el nombre màxim de nivells [17].*

Per acabar, a la Taula 3.1 podem veure els resultats d'aplicar els diferents kernels que hem vist anteriorment als exemples. Notem que hem normalitzat els kernels segons la Definició 23. Observem que, llevat del primer exemple, el valor del Node Histogram Kernel val 1 per tenir tots els grafs dels exemples el mateix nombre d'etiquetes de cada tipus.

	Node Histogram	Shortst Path	Weisfeiler Lehman	ODD-sth	Pyramid Match
Exemple 28	0.992	0.827	0.150	0.756	0.58
Exemple 33	1	0.73	0.13	0.97	0.526
Exemple 38	1	0.944	0.197	0.901	0.611
Exemple 39	1	0.744	1	0.558	0.717
Exemple 41	1	0.956	0.237	0.772	0.646

Taula 3.1: Vegem els valors normalitzats obtinguts al aplicar-los als grafs dels exemples vists al llarg del capítol.

CAPÍTOL 4

APLICACIÓ DELS KERNELS ALS M-DAGS

En aquest capítol aplicarem els kernels de grafs definits al Capítol 3 als m-DAGs d'un conjunt d'organismes. Abans però, per tal de poder interpretar millor els resultats obtinguts en l'aplicació dels kernels, realitzarem un estudi topològic dels m-DAGs.

4.1 Anàlisi topològic dels m-DAGs

Al nostre estudi farem us d'un data set format per 436 organismes de la base de dades de KEGG [4, 5]. Comptam amb 150 organismes del regne animal, 48 plantes, 47 fongs, 155 bacteris i 36 arquees. La llista dels organismes seleccionats es pot consultar a la llista d'organismes adjuntada a la carpeta del treball (llista_organismes). Per a cada un dels organismes, amb l'eina MetaDAG [9] obtenim els corresponents m-DAGs a partir dels reaction graphs. Disposam doncs de 436 m-DAGs dels que estudiarem la seva topologia (connectivitat, densitat, etc.) i també la seva composició (grandària, densitat, etc. de les MBBs). Farem aquest estudi mitjançant el paquet *igraph* [19] de Python.

Pel que fa a la connectivitat dels m-DAGs, a la Figura 4.1 es mostra el nombre de components connexes de cada m-DAG. Com podem observar, gairebé tots els m-DAGs tenen un nombre de components connexes bastant gran. En el cas dels animals, els m-DAGs tenen entre 150 i 225 components connexes. Mentre que els bacteris i les arquees tenen un nombre de component connexes molt variable.

Respecte a cada component connexa, per tal d'analitzar com es distribueixen els nodes entre totes aquestes components connexes, hem calculat l'ordre de cada una per a cada m-DAG (Figura 4.2). Notem que gairebé tots els m-DAGs tenen una component connexa considerablement més gran que totes les altres, i que, a més, les altres components connexes tenen un nombre de nodes molt baix. De fet, en general les mitjanes de nodes per component connexa dels m-DAGs es troben entre 1.35 i 6.64 però el nombre de nodes de la component connexa més gran és major de 320 per més de la mitat dels m-DAGs. Per tant, sembla interessant estudiar aquesta component connexa en més detall. També cal destacar que el nombre de components connexes

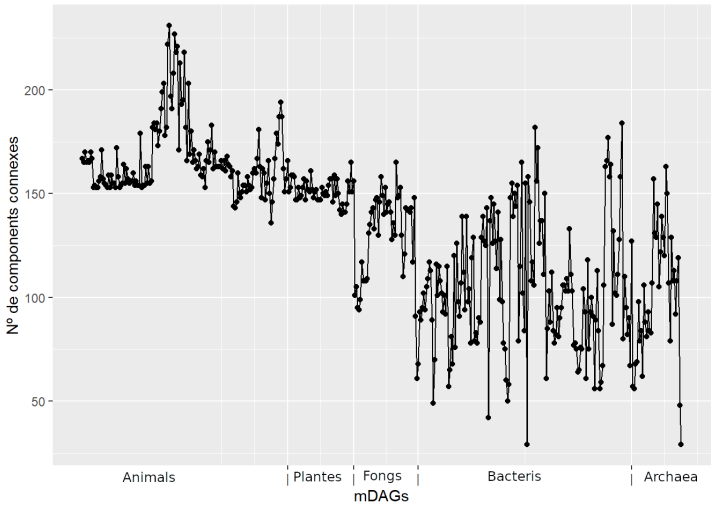


Figura 4.1: Distribució del nombre de components connexes dels m-DAGs del nostre data set.

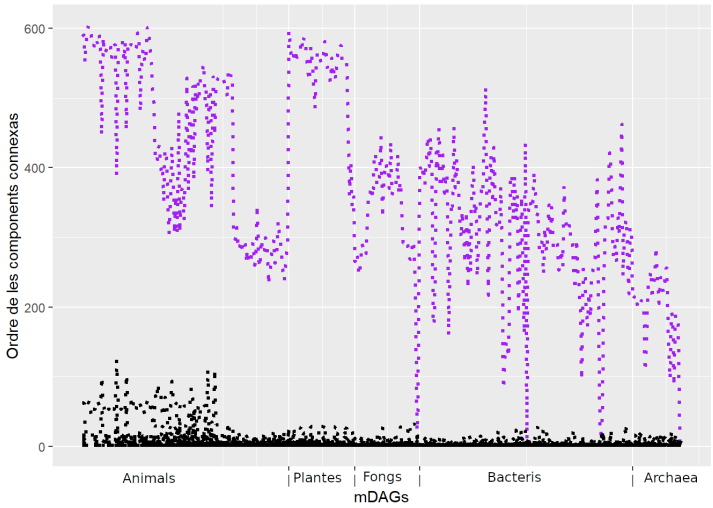


Figura 4.2: Els ordres de les components connexes per cada m-DAG. La component d'ordre major de cada m-DAG està en color violeta i la resta de components en negre.

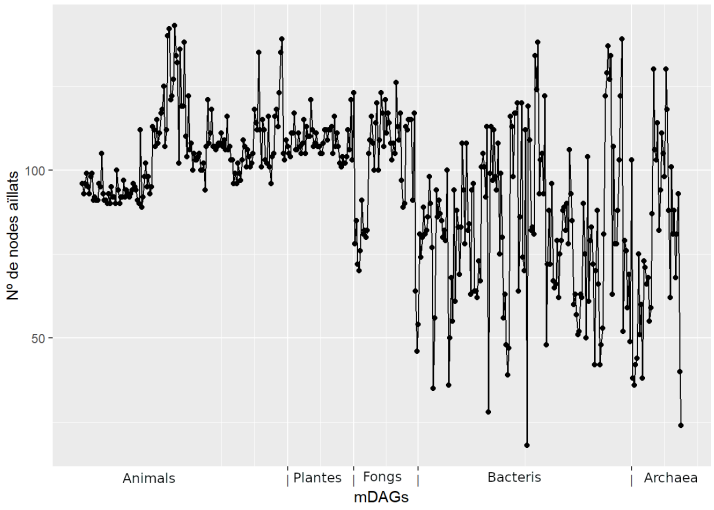


Figura 4.3: Distribució de nodes aïllats per a cada m-DAG.

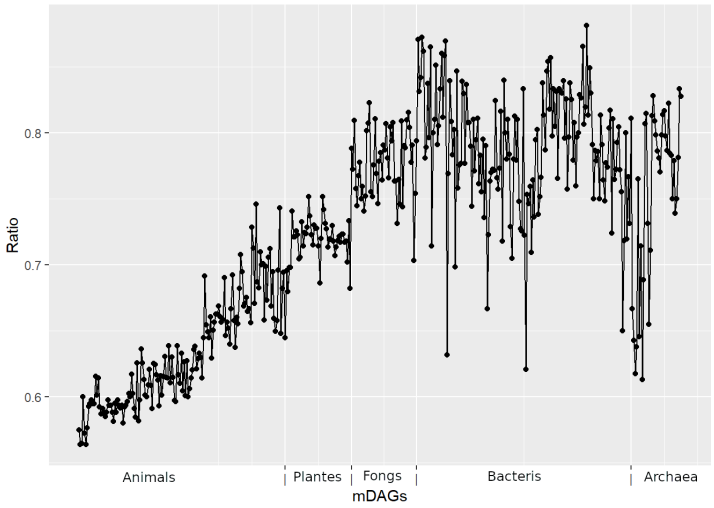


Figura 4.4: Distribució de la ràtio de nodes aïllats respecte al nombre de components connexes per a cada m-DAG.

amb un sol node es troba entre 18 i 143. La Figura 4.3 mostra el nombre de nodes aïllats per a cada m-DAG, agrupats per regne. Com podem veure, els animals, que tenen major nombre de components connexes, també tenen major nombre de nodes aïllats. En efecte, en aquesta figura s'observa que el nombre de nodes aïllats dels m-DAGs segueix un patró molt similar al node de components connexes. De fet, si consideram la ràtio de nodes aïllats respecte al nombre de components connexes de cada m-DAG, tal com es mostra a la Figura 4.4, tenim que aquestes ràtios es troben entre 0.55 i 0.9. Com veim a la figura, les ràtios dels m-DAGs de fongs, bacteris i arquees són més alts de mitjana que els Animals i les Plantes, el que significa que tenen més nodes aïllats en proporció al seu nombre de components connexes de mitjana.

Respecte a la composició dels m-DAGs, en particular, al nombre de reaccions que té cada MBB, a la Figura 4.5 podem observar que les MBBs de tots els m-DAGs del nostre

data set, llevat de la MBB més gran, estan formades per menys de 100 reaccions. A més, la MBB més gran de cada m-DAG pertany a la component connexa de major ordre per a tots els m-DAGs del data set. Així doncs, podem concloure que la component de major ordre de cada m-DAG emmagatzema quasi tota la informació metabòlica.

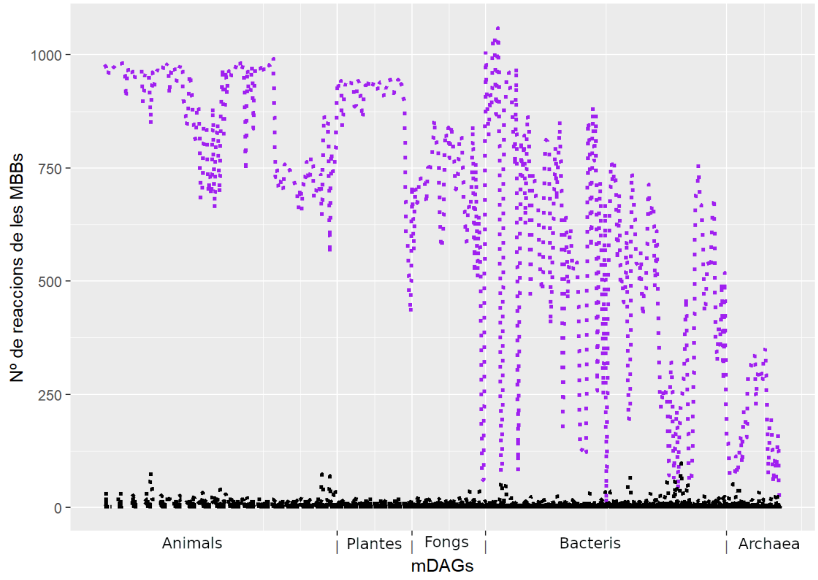


Figura 4.5: A la figura podem observar el nombre de reaccions de les MBBs dels diferents m-DAGs del nostre data set ordenats per regnes i marcat en violeta la MBB més gran de cada m-DAG i en negre la resta.

A continuació desglossam l'estudi fet anteriorment agrupat per regnes per veure si hi ha diferències significatives. A la Taula 4.1 observam que la mitjana del nombre de components connexes dels m-DAGs i dels ordres de les dues components connexes de major ordre dels animals és molt més alta que als altres regnes.

Regne	mitjana nº ccs	mitjana nº nodes aïllats	mitjana ordres ccs major	mitjana ordres 2a ccs major
Animal	231	143	604	125
Planta	165	123	597	28
Fong	165	126	445	32
Bacteri	184	139	516	29
Arquea	163	130	286	20

Taula 4.1: A la taula podem veure informació sobre les components connexes (ccs) dels m-DAGs en funció del regne.

Al gràfic de violins de la Figura 4.6 podem veure que l'ordre de la component connexa més gran és, en general, major als m-DAGs dels animals i les plantes. Els fongs i els bacteris varien molt i les arquea sempre és menor de 300.

Respecte a la densitat dels m-DAGs, acabam de veure que tots els m-DAGs tenen una component connexa molt més gran que la resta, i que també tenen moltes components connexes petites. Per tant, estudiarem la densitat d'aquesta component connexa gran. A la Figura 4.7 es mostra la distribució de graus d'entrada i de sortida dels nodes de la component connexa més gran de cada m-DAG. Com podem observar, es repeteix el mateix patró observat a l'estudi de la connectivitat, és a dir, hi ha un node (una

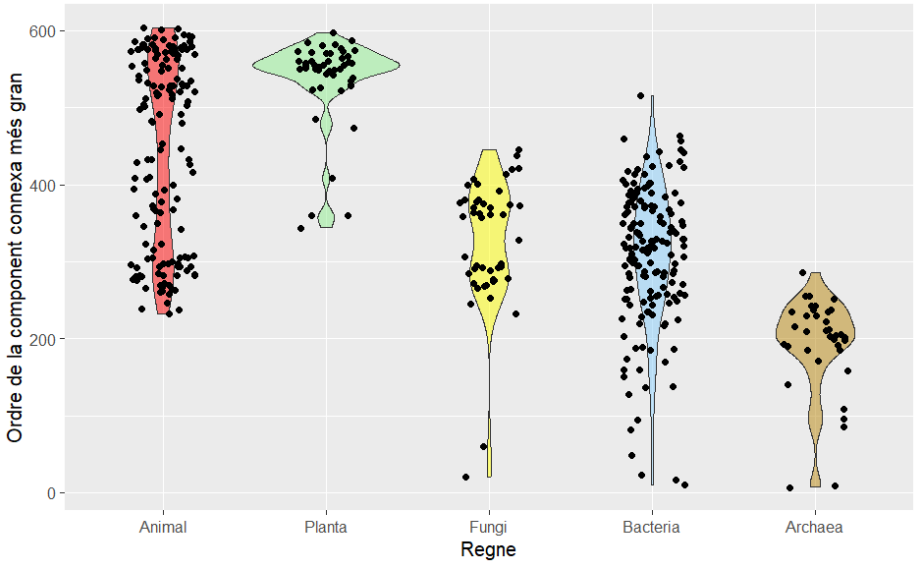


Figura 4.6: A la figura mostram els ordres de les components connexes més granss de cada m-DAG segons el regne al qual pertany.

MBB) molt més connectat que la resta. En efecte, mirant els punts pintats en negre a la figura, podem observar que el grau dels nodes baixa considerablement llevat del node de grau major en la majoria dels m-DAGs. Així doncs, en resum tenim que, els m-DAGs dels nostre data set tenen una component connexa gran i molts de nodes aïllats i components connexes molt petites. Dins cada component connexa gran, hi ha una MBB que té un nombre de reaccions molt gran i la resta en tenen molt poques, i, per acabar, aquesta MBB que té moltes reaccions també té grau d'entrada i sortida molt més gran que la resta, que pràcticament tenen grau 1.

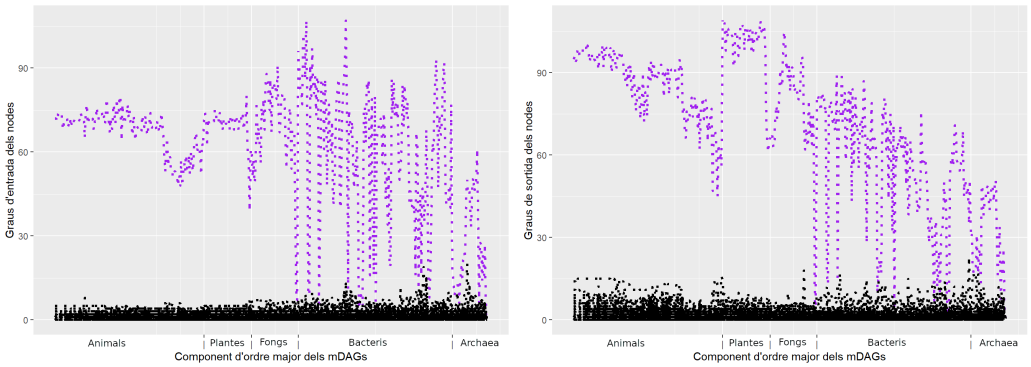


Figura 4.7: Graus de sortida i d'entrada de la component d'ordre major de cada m-DAG, respectivament. El node de major grau de cada m-DAG està de color violeta i la resta de color negre.

4.2 Kernels de grafs per comparar m-DAGs

En aquesta secció estudiarem el comportament dels diferents kernels de grafs aplicats als m-DAGs del nostre data set. Per això, consideram totes les parelles de m-DAGs del nostre data set i els següents kernels: Node Histrogram, Shortest Path, Weisfeiler Lehman, ODD-sth i Pyramid Match. Cada un d'aquests kernels considera característiques diferents dels grafs a comparar i, el que tenen tots en comú, és que es poden aplicar a grafs etiquetats i dirigits. En el nostre cas, a més de dirigits els grafs són acíclics, per la qual cosa creim molt interessant considerar un kernel específic per aquest tipus de grafs com és el ODD-sth kernel. Per calcular el valor de similitud de cada parella de m-DAGs per a cada un dels kernels escollits, usam el paquet *grakel* [20] de Python. Obtenim així una matriu quadrada de 436 files que emmagatzema els valors d'aquests kernels. Posteriorment, normalitzarem la matriu de la forma vista a la Definició 23. Finalment, per visualitzar cada una de les matrius obtingudes, considerarem la seva representació en forma de *heatmap*. Un heatmap, o mapa de color, permet visualitzar les matrius de similaritat de forma gràfica. Cada element en la fila i i columna j del heapmap representa el valor del kernel entre els m-DAGs i-èssim i j-èssim del nostre data set, i en funció d'aquest valor se li assigna un color d'entre una escala de colors. Observarem que els elements de la diagonal són tots del mateix color corresponent al valor més alt, el groc, i que el heatmap és simètric respecte de la diagonal per ser la matriu del kernel una matriu simètrica.

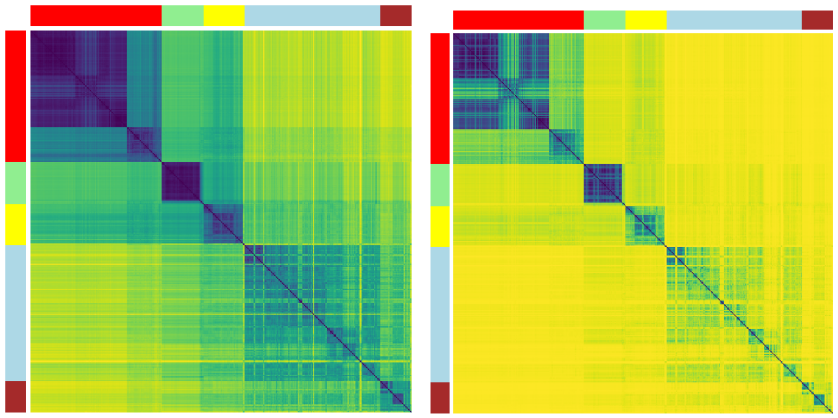
A la Figura 4.8 es mostren els 5 heatmaps obtinguts d'aplicar al nostre data set els kernels del Capítol 3. Observem que tots presenten el mateix patró. Notem que el kernel més senzill i menys costos, el Node Histogram Kernel, funciona molt bé per al nostre data set perquè, com hem vist abans, tots els m-DAGs tenen una component connexa d'ordre molt alt i les altres molt petites, és a dir, no tenen quasi estructura i, per tant, mesurar la distància tenint en compte només les etiquetes dels nodes ens dona molta informació.

Per tal d'avaluar els resultats obtinguts amb els diferents kernels, considerarem la classificació taxonòmica a nivell de regne dels organismes dels quals hem obtingut els m-DAGs. És a dir, cada m-DAG queda classificat en un dels següents tipus: Animal, Planta, Fong, Bacteri o Arquea.

D'altra banda, obtindrem una classificació a partir dels valors obtinguts per cada kernel mitjançant un mètode de clustering. El *clustering* és un algoritme que té com a objectiu agrupar les dades en conjunts d'objectes no etiquetats en funció de la similitud de les seves característiques, aquests conjunts s'anomenen clústers. Hi ha molts de mètodes de clustering diferents, a continuació definirem el Clustering Espectral que és el que farem servir.

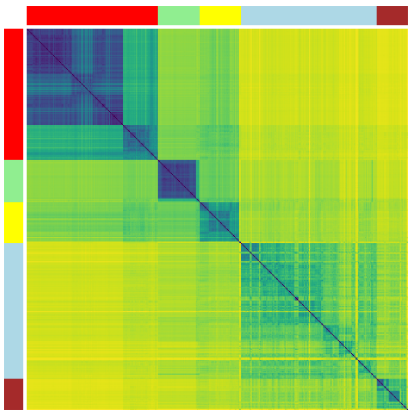
Els mètodes de *Clustering Espectral* consideren la matriu laplaciana de la matriu de similitud de les dades i fan ús de l'espectre d'aquesta (valors i vectors propis) per fer una reducció de dimensionalitat per després agrupar les dades representades a un espai de dimensió menor mitjançant un mètode de clustering que consideri les dades en forma de punts. Existeixen diferents mètodes de clustering espectral en funció de la definició de matriu Laplaciana escollida, el paquet *scikit-learn* [21] de Python que usarem defineix la matriu laplaciana d'una matriu de similitud S com

$$L_{norm} = I - D^{-1/2} S D^{-1/2}$$

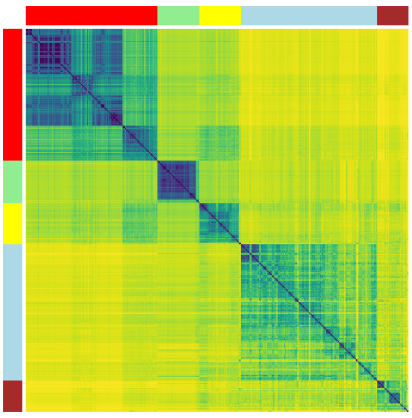


(a) NodeHistogram

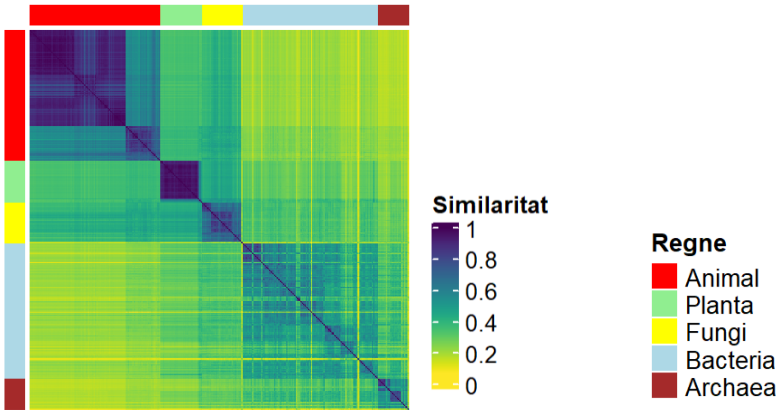
(b) ShortestPath



(c) WeisfeilerLehman



(d) ODDsth



(e) PyramidMatch

Figura 4.8: Heatmaps de les matrius normalitzades dels kernels obtinguts.

on $D = (d_{ij})$ és una matriu diagonal on $d_{ii} = \sum_{j=1}^{n} s_{ij}$, és a dir, cada element d_{ii} és la suma de tots els elements de la fila i a la matriu S i I és la matriu identitat [22].

Un cop obtinguts els punts, farem servir el mètode de *Clustering K-means* per agrupar-los. Aquest mètode cerca una partició del conjunt d'objectes en un nombre fix k de clústers els quals s'identifiquen mitjançant els seus punts mitjans [23]. Donats els punts $x_1, \ldots, x_p \in \mathbb{R}^n$, l'objectiu és trobar els k punts $c_1, \ldots, c_p \in \mathbb{R}^n$ que minimitzin

$$SS_c(x_1, \ldots, x_p; k) = \sum_{i=1}^{p} \min_{j=1,\ldots,k} ||x_i - c_j||^2$$

on $||\cdot||$ és la norma euclidiana. Llavors, cada c_j definirà un clúster format pels punts x_i que es troben més a prop d'ell que de qualsevol altre c_m. Formalment, c_j definirà el clúster

$$C_j = \{x_i : ||x_i - c_j|| < ||x_i - c_m|| \forall\, m \neq j\}.$$

A la Taula 4.2 mostram el nombre d'elements de cada regne que pertanyen a cada clúster de l'agrupació que resulta d'aplicar Clustering Espectral a les matrius de kernels (similitud) calculades anteriorment. Hem considerat $K = 5$ clústers perquè tenim 5 regnes a la classificació taxonòmica.

(a) NodeHistogram

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	116	0	0	0	0
2	0	44	0	0	0
3	34	4	47	0	0
4	0	0	0	152	0
5	0	0	0	3	36

(b) ShortestPath

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	150	1	0	0	0
2	0	47	0	0	0
3	0	0	45	0	0
4	0	0	2	155	12
5	0	0	0	0	24

(c) WeisfeilerLehman

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	116	0	0	0	0
2	0	44	0	0	0
3	34	4	47	0	0
4	0	0	0	155	0
5	0	0	0	0	36

(d) ODDsth

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	111	0	0	0	0
2	0	44	0	3	0
3	39	4	47	3	1
4	0	0	0	152	2
5	0	0	0	0	33

(e) PyramidMatch

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	110	0	0	0	0
2	0	44	0	0	0
3	40	4	47	0	0
4	0	0	0	154	0
5	0	0	0	1	36

Taula 4.2: Les taules representen els clústers que resulten d'aplicar clustering espectral a cada una de les matrius de similitud i la relació d'elements de cada regne que hi ha a cada clúster.

Ara, per tal d'avaluar els resultats dels kernels primer assignarem un regne a cada cluster usant l'algorisme de Maximum Weight Matching (Secció 2.1). Considerem el conjunt de clústers $C = \{C_i | 1 \leq i \leq 5\}$ i el conjunt de regnes $R = \{R_i | 1 \leq i \leq 5\}$. Sigui el graf bipartit complet $G = (V, E)$ amb $\{C, R\}$ la partició del conjunt de nodes i definim la funció de pesos de les arestes com $w : E \to \mathbb{R}$ on $w(C_i R_j)$ és el nombre d'elements que tenen en comú el clúster C_i i el regne R_j. D'ara en endavant, considerarem el matching M resultant d'aplicar Maximum Weight Matching al graf G.

Donat un clúster C_i i R_j el regne que li correspon segons M, considerarem:

- VP (verdaders positius) com el nombre de m-DAGs que s'han classificat a C_i i pertanyen a R_j
- FP (falsos positius) com el nombre de m-DAGs que s'han classificat a C_i i no pertanyen a R_j
- VN (vertaders negatius) com el nombre de m-DAGs que no pertanyen a R_j i no s'han classificat a C_i.
- FN (falsos negatius) com el nombre de m-DAGs que pertanyen a R_j i no s'han classificat a C_i.

A continuació, introduïm aquí els següents conceptes per a l'avaluació de les prediccions.

- La *precisió* mesura la qualitat de les prediccions positives. És la proporció de vertaders positius (VP) entre totes les instàncies classificades com a positives (tant vertaders positius com falsos positius, FP)

$$presició = \frac{VP}{VP+FP}$$

- El *recall*, també conegut com a sensibilitat, es centra en la capacitat per identificar totes les instàncies positives. És la proporció de vertaders positius (VP) entre totes les instàncies que són realment positives (la suma de vertaders positius i falsos negatius, FN).

$$recall = \frac{VP}{VP+FN}$$

- El *F1 Score* és útil quan hi ha un desequilibri entre les classes i es vol una mètrica que combini la precisió i l'exhaustivitat. El F1 Score es defineix com la mitjana harmònica de la precisió i el recall.

$$F1 = \frac{2 \cdot presició \cdot recall}{presició + recall} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$$

- L'*accuracy* o exactitud mesura la proporció de prediccions correctes (tant vertaders positius com vertaders negatius) respecte al total de prediccions realitzades. És una mesura global del rendiment del model.

$$accuracy = \frac{VP+VN}{VP+VN+FP+FN}$$

A la Taula 4.3 es mostren els resultats obtinguts per cada kernel. Notam que el kernel que millor classifica per regnes és el ShortestPath seguit pel Weisfeiler Lehman i el NodeHistogram amb tots els paràmetres d'avaluació majors que 0.9. A la Taula 4.4 podem consultar els temps d'execució en segons de cada kernel. Observem que, segons els costs computacionals definits a la Secció 3, el menys costos dels kernels és el Node Histogram seguit del Weisfeiler Lehman. El Shortest-Path i el ODD-sth són els més costosos en diferència, el que pot resultar en intractables quan el data set és molt gros.

	NodeHist	ShortestPath	WeisfeilerLehman	ODD-sth	PyramidMatch
Precisió	0.945	0.968	0.952	0.941	0.946
Recall	0.906	0.966	0.913	0.888	0.897
F1 score	0.913	0.963	0.92	0.899	0.906
Accuracy	0.906	0.966	0.913	0.888	0.897

Taula 4.3: Accuracy i mitjana ponderada de la precisió, recall i f1 score dels clústers de la Figura 4.2

	nodehistogram	W-L	shortest-path	ODD-sth	pyramid match
temps(s)	0.25	3.31	110.86	158.73	377.08

Taula 4.4: Temps d'execució en segons de les matrius de kernels.

4.3 Nous kernels a partir de les observacions

Com hem vist a la Secció 4.1 els m-DAGs del nostre data set tenen una component connexa d'ordre molt alt comparat amb la resta de les components del graf i la resta amb ordres molt baixos. Notem que, a l'hora de comparar aquests m-DAGs, l'estructura de la component connexa d'ordre major és rellevant però l'estructura de la resta no és significativa. Llavors, podem crear un nou kernel adaptat al tipus de grafs que estam tractant. Primer, dividirem el graf en dos: la component connexa de major ordre i la resta de subgrafs. La idea consisteix en aplicar el kernel que s'adapti millor a cada un dels grafs en els quals hem dividit el graf original. Els grafs formats per les components connexes de menor ordre els podem comparar usant el Node Histogram Kernel perquè és un kernel que funciona bé per grafs amb poca estructura (poques arestes). Per comparar la component connexa de major ordre dels m-DAGs podem usar el Weisfeiler Lehman, el Shortest Path o el ODD-sth Kernel que són els que han obtingut millors resultats a l'hora de classificar per regnes a la Secció 4.2. El primer té en compte les etiquetes i l'estructura, el segon compara les distàncies entre els nodes tenint en compte les etiquetes i el tercer destaca per adaptar-se millor a l'estructura de grafs dirigits.

Donat $\mathcal{G}$ un conjunt finit de grafs, denotarem per $C(\mathcal{G})$ els conjunts de les components connexes d'ordre major de $\mathcal{G}$ i $R(\mathcal{G})$ el conjunt dels subgrafs formats per la resta de les components connexes de cada graf de $\mathcal{G}$.

Primer, veurem els resultats que obtenim al sumar el valor d'aquests kernels, donant-li un pes determinat a cada un.

Definició 58 (Kernel Suma) *Donats els kernels K_1 a $C(\mathcal{G}) \times C(\mathcal{G})$ i K_2 a $R(\mathcal{G}) \times R(\mathcal{G})$, dos grafs $G, G' \in \mathcal{G}$ i siguin $(c, c') \in C(\mathcal{G}) \times C(\mathcal{G})$ les seves components connexes de major ordre i $(r, r') \in R(\mathcal{G}) \times R(\mathcal{G})$ els subgrafs format per la resta de components connexes, respectivament. Definim el Kernel Suma com $K_{suma}(G, G') = \alpha K_1(c, c') + (1-\alpha)K_2(r, r')$ per $\alpha \in (0, 1)$.*

Proposició 59 *El Kernel Suma és un kernel.*

Prova. Notem que, per ser $\alpha > 0$, $\alpha K_1(c, c')$ és un kernel a $C(\mathcal{G}) \times C(\mathcal{G})$ i $(1-\alpha)K_2(r, r')$ és un kernel a $R(\mathcal{G}) \times R(\mathcal{G})$ per la Proposició 13. Ara, la suma d'aquests, $K_{suma}(G, G')$ és un kernel a $(C(\mathcal{G}) \times R(\mathcal{G})) \times (C(\mathcal{G}) \times R(\mathcal{G}))$ per la Proposició 14.

□

Per tal de trobar el valor α òptim, calcularem la matriu de kernels per a diferents valors de α. Notem que tant pel nou kernel amb WL (Taula 4.6) com pel nou kernel amb SP (Taula 4.5) obtenim millors resultats per valors d'alfa majors (de fet, propers a 1), és a dir, quan el kernel entre les components connexes d'ordre major té més pes al nou kernel. D'aquest fet podem deduir que les components connexes de major ordre de cada m-DAG són molt rellevants a l'hora d'estudiar-los i comparar-los.

	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
Precisió	0.916	0.916	0.916	0.915	0.915	0.92	0.938	0.942	0.947
Recall	0 0.851	0.851	0.851	0.849	0.849	0.86	0.888	0.899	0.92
F1	0.864	0.864	0.864	0.862	0.862	0.872	0.897	0.907	0.925
Accuracy	0.851	0.851	0.851	0.849	0.849	0.86	0.888	0.899	0.92

Taula 4.5: Valors dels paràmetres d'avaluació considerant diferents valors de α per calcular el nou kernel amb $K1$ Shortest Path i $K2$ Node Histogram.

	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
Precisió	0.917	0.922	0.928	0.934	0.936	0.94	0.942	0.945	0.949
Recall	0.853	0.865	0.876	0.885	0.888	0.892	0.897	0.901	0.908
F1	0.866	0.876	0.886	0.895	0.897	0.901	0.905	0.91	0.916
Accuracy	0.853	0.865	0.876	0.885	0.888	0.892	0.897	0.901	0.908

Taula 4.6: Valors dels paràmetres d'avaluació considerant diferents valors de α per calcular el nou kernel amb $K1$ Weisfeiler Lehman i $K2$ Node Histogram.

	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
Precisió	0.922	0.926	0.934	0.936	0.936	0.942	0.944	0.948	0.948
Recall	0.865	0.872	0.883	0.885	0.885	0.892	0.897	0.901	0.901
F1	0.876	0.882	0.893	0.895	0.895	0.901	0.905	0.91	0.91
Accuracy	0.865	0.872	0.883	0.885	0.885	0.892	0.897	0.901	0.901

Taula 4.7: Valors dels paràmetres d'avaluació considerant diferents valors de α per calcular el nou kernel amb $K1$ ODD-sth i $K2$ Node Histogram.

Una altra forma de definir el nou kernel és fent el producte d'un kernel aplicat a les components connexes de major ordre i d'un kernel aplicat al subgrafs formats per la resta de components connexes. Ho definirem formalment a continuació.

Definició 60 (Kernel Producte) *Donats els kernels K_1 a $C(\mathcal{G}) \times C(\mathcal{G})$ i K_2 a $R(\mathcal{G}) \times R(\mathcal{G})$, dos grafs $G, G' \in \mathcal{G}$ i siguin $(c, c') \in C(\mathcal{G}) \times C(\mathcal{G})$ les seves components connexes de major ordre i $(r, r') \in R(\mathcal{G}) \times R(\mathcal{G})$ els subgrafs format per la resta de components connexes, respectivament. Definim el Kernel Producte com $K_{producte}(G, G') = K_1(c, c')K_2(r, r')$.*

Proposició 61 *El Kernel Producte és un kernel de grafs.*

Prova. Notem que $K_{producte}$ és un kernel a $\mathcal{G} \times \mathcal{G}$ per ser un R-Convolutional kernel (Definició 18) on S i S' contenen dos conjunts de subestructures($d = 2$) que són el

conjunt de les components connexes de major ordre i el conjunt dels subgrafs formats per la resta de les components connexes de G i G' i els kernels són K_1 i K_2.

□

Notem, observant la Taula 4.9, que els paràmetres del nou kernel amb el Weisfeiler Lehman i ODD-sth són molt bons (tots més alts de 0.98). En canvi, el nou kernel amb Shortest path no funciona massa bé. En efecte, a la Taula 4.8 podem veure que SP junta tots els animals, plantes i fongs en un clúster mentre que la resta de kernels obtenen clústers pràcticament homogenis de cada regne.

(a) k1:Shortest Path

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	150	48	45	0	0
2	0	0	0	2	0
3	0	0	2	0	0
4	0	0	0	153	14
5	0	0	0	0	22

(b) k1:Weisfeiler-Lehman

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	150	4	1	0	0
2	0	44	0	0	0
3	0	0	45	0	0
4	0	0	1	155	0
5	0	0	0	0	36

(c) k1:ODD-sth

Clúster	Animal	Planta	Fong	Bacteria	Arquea
1	150	4	1	0	0
2	0	44	0	0	0
3	0	0	45	0	0
4	0	0	1	155	0
5	0	0	0	0	36

Taula 4.8: Les taules representen els clústers que resulten d'aplicar Clustering Espectral a cada una de les matrius de similitud dels nous kernels on K_2 és el Node Histogram Kernel i K_1 el Shortest path, Weisfeiler Lehman o ODD-sth kernel.

	K1:ShortestPath	K1:Weisfeiler-Lehman	K1:ODD-sth
Precisió	0.728	0.987	0.987
Recall	0.75	0.986	0.986
F1 score	0.672	0.986	0.986
Accuracy	0.75	0.986	0.986

Taula 4.9: La taula mostra accuracy i la mitjana ponderada de la precisió, recall i f1 score dels clústers de la Taula 4.8

Finalment, el Kernel Producte amb WL sembla ser el més eficient per classificar m-DAGs perquè té un cost computacional molt menor que el ODD-sth Kernel, el qual és una característica important en cas de tenir un data set nombrós.

CAPÍTOL 5

CONCLUSIONS

En aquest treball hem analitzat l'aplicació dels kernels de grafs per a la comparació i classificació del model m-DAG de xarxes metabòliques.

Per això, primer hem introduït els fonaments matemàtics necessaris per a definir els kernels (Secció 2) i per poder demostrar que, efectivament ho són (Capítol 3). Aquesta ha resultat ser una de les tasques més difícils d'aquest treball, ja que molts dels articles consultats no expliquen de forma clara els resultats i, per aquesta raó, hem hagut de consultar un gran nombre d'articles diferents per poder entendre i demostrar el funcionament dels kernels.

Posteriorment, hem seleccionat un conjunt de prova (data set) d'organismes de KEGG [4, 5] per tal d'avaluar el comportament dels kernels escollits i hem obtingut els m-DAGs corresponents a cada organisme amb l'eina metaDAG [9]. El data set de prova conté 150 organismes del regne animal, 48 plantes, 47 fongs, 155 bacteris i 36 archaeas, el que fa un total de 436 organismes taxonòmicament prou diferenciats.

Després d'estudiar les característiques topològiques del nostre data set, podem concloure que els m-DAGs d'aquest tenen una component connexa principal d'ordre molt gran i un nombre elevat de components connexes molt petites, moltes de les quals són nodes aïllats. Hem vist aquest fet reflectit als resultats d'aplicar els kernels als m-DAGs del data set. El Node Histogram Kernel, tot i ser el més senzill, ha proporcionat bons resultats pels m-DAGs, degut al gran nombre de nodes aïllats que caracteritzen aquestes xarxes. Cal recalcar que el Shortest Path i el Weisfeiler-Lehman Kernel han estat els més efectius al nostre data set degut a la seva capacitat de capturar la connectivitat i les estructures complexes dels m-DAGs (Secció 4.2).

Per acabar, hem creat nou kernels adaptats a les caracteristiques de nostre data set: Kernel Suma i Kernel Producte. Cal destacar que el Kernel Producte amb el Weisfeiler-Lehman i ODD-sth Kernel (Secció 4.3) ha obtingut resultats vertaderament bons, però el fet que ODD-sth és computacionalment molt costós fa que ens decantem pel Kernel producte amb Weisfeiler Lehman com el més eficient per classificar m-DAGs.

CAPÍTOL 6

APÈNDIX A

A la carpeta que s'adjunta amb aquesta memòria es pot trobar:

- fitxer amb la llista dels organismes seleccionats ordenats per regnes (llista_organismes).
- fitxer python amb el codi per crear el diccionari amb la informació topològica dels m-DAGs (info_topologica_mDAGs.py).
- fitxer python amb diccionari amb la informació topològica dels m-DAGs (dict_info_topologica_mDAGs.py).
- fitxer python amb codi per comprovar que la MBB més gran de cada m-DAG pertany a la component connexa de major ordre (mbb1_pertany_a_css1.py).
- matriu obtinguda al aplicar el Node Histogram Kernel als m-DAGs (NH).
- matriu obtinguda al aplicar el Shortest Path Kernel als m-DAGs (SP).
- matriu obtinguda al aplicar el Weisfeiler Lehman Kernel als m-DAGs (WL).
- matriu obtinguda al aplicar el ODD-sth Kernel als m-DAGs (ODD).
- matriu obtinguda al aplicar el Pyramid Match Kernel als m-DAGs (PM).
- fitxer python amb el codi pel clustering dels kernels (clustering_kernels.py).
- fitxer python amb el codi dels nous kernels i el clustering d'aquests (nous_kernels.py).

BIBLIOGRAFIA

[1] R. Alberich, J. A. Castro, M. Llabres, and P. Palmer-Rodriguez, "Metabolomics analysis: Finding out metabolic building blocks," *PloS one*, vol. 12, no. 5, p. e0177031, 2017. 1, 2.2

[2] K. Borgwardt, E. Ghisu, F. Llinares-López, L. O'Bray, B. Rieck *et al.*, "Graph kernels: State-of-the-art and future challenges," *Foundations and Trends® in Machine Learning*, vol. 13, no. 5-6, pp. 531–712, 2020. 1, 2.3, 2.3.1, 3.1, 3.1.2

[3] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis, "Graph kernels: A survey," *Journal of Artificial Intelligence Research*, vol. 72, pp. 943–1027, 2021. 1, 2.3.1

[4] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and minimoru Kanehisa, "KEGG: Kyoto Encyclopedia of Genes and Genomes," *Oxford University Press*, vol. 28, no. 1, pp. 27–30, 2000. 1, 2.2, 4.1, 5

[5] dagimoru Kaneisha, S. Goto, Y. Sato, M. Kawashima, M. Furumichi, and M. Tanable, "KEGG: Data, information, knowledge and principle: back to metabolism in KEGG," *Nucleic Acids Research*, vol. 42, no. D1, pp. D199–D205, 2014. 1, 2.2, 4.1, 5

[6] D. Haussler, "Convolution kernels on discrete structures," Citeseer, Tech. Rep., 1999. 1, 2.3.1, 2.3.1, 2.3.1, 2.3.1

[7] J. C. Pons Mayol, "Teoria de grafs," in *Apunts Matemàtica Discreta*, 2020. 2.1

[8] Hungarian algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Hungarian_algorithm 2.1

[9] P. Palmer-Rodríguez, R. Alberich, M. Reyes-Prieto, J. A. Castro, and M. Llabrés, "Metadag: a web tool to generate and analyse metabolic networks," *bioRxiv*, 2024. [Online]. Available: https://www.biorxiv.org/content/early/2024/05/17/2024.05.15.593827 2.2, 4.1, 5

[10] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950. 2.3.1

[11] Schur product theorem. [Online]. Available: https://en.wikipedia.org/wiki/Schur_product_theorem#Proof_of_positive_semidefiniteness 2.3.1

[12] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE international conference on data dagidagg (ICDM'05)*. IEEE, 2005, pp. 8–pp. 3.1.1, 3.1.1

[13] Algoritmo de dijkstra. [Online]. Available: https://es.wikibooks.org/wiki/Algoritmia/Algoritmo_de_Dijkstra 3.1.1

[14] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011. 3.1.2, 3.1.2

[15] G. Da San Martino, N. Navarin, and A. Sperduti, "Tree-based kernel for graphs with continuous attributes," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3270–3276, 2017. 46

[16] F. Aiolli, G. Da San Martino, A. Sperduti, and A. Moschitti, "Fast on-line kernel learning for trees," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 787–791. 3.1.3

[17] K. Grauman and T. Darrell, "The pyramid match kernel: Efficient learning with sets of features." *Journal of Machine Learning Research*, vol. 8, no. 4, 2007. 3.1.4, 3.1.4, 57

[18] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 31, no. 1, 2017. 3.1.4

[19] igraph. [Online]. Available: https://github.com/igraph/python-igraph.git 4.1

[20] Grakel. [Online]. Available: https://github.com/ysig/GraKeL.git 4.2

[21] scikit-learn. [Online]. Available: https://github.com/scikit-learn/scikit-learn.git 4.2

[22] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007. 4.2

[23] I. García, R. Alberich, A. Mir, F. Roselló, and M. Oliva. Análisis de datos. [Online]. Available: https://aprender-uib.github.io/AD/ 4.2