Algoritmo de selección de instancias para bases de datos de grafos

Por:

Magdiel Jiménez Guarneros

Tesis sometida como requisito parcial
para obtener el grado de:

MAESTRÍA EN CIENCIAS EN EL ÁREA DE CIENCIAS COMPUTACIONALES

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Tonantzintla, Puebla

Dirigida por:

Dr. Jesús Ariel Carrasco Ochoa
Dr. José Francisco Martínez Trinidad

Resumen

La clasificación supervisada de grafos es una actividad que ha generado un gran interés en diversos dominios de aplicación de las ciencias y la ingeniería. Esta actividad tiene como propósito, construir un modelo o clasificador que pueda ser utilizado para determinar la clase a la que pertenecen nuevos grafos, usando un conjunto de grafos de entrenamiento previamente etiquetados. Sin embargo, al igual que ocurre con datos n-dimensionales, no todos los grafos de un conjunto de entrenamiento son de utilidad para la clasificación, resultando indispensable un proceso para descartar aquellos grafos innecesarios. Este proceso es conocido como *selección de instancias* y su principal objetivo radica en reducir el tamaño del conjunto de entrenamiento, manteniendo, tanto como sea posible, la misma calidad de clasificación que el conjunto original. En la literatura, no se reportan algoritmos para la selección de instancias en bases de datos de grafos, pues los algoritmos existentes han sido desarrollados para espacios n-dimensionales.

En este trabajo de investigación abordamos el problema de la selección de instancias en bases de datos de grafos siguiendo tres diferentes enfoques. En el primer enfoque, se realiza una selección de instancias en el espacio de los grafos, usando una medida de disimilaridad diseñada para este espacio de representación. En el segundo enfoque, la selección de instancias se lleva a cabo en el espacio n-dimensional generado por el *embedding de grafos vía disimilaridad*. Finalmente, en el tercer enfoque se desarrolla una selección de instancias híbrida mediante la combinación de los dos enfoques anteriores.

En particular, en el espacio de los grafos se proponen tres algoritmos para la

selección de instancias. El primero de estos algoritmos descarta aquellos grafos del conjunto de entrenamiento que no afectan la calidad de clasificación, procesando cada grafo en orden de la disimilaridad promedio que tiene éste con respecto a los demás grafos de su clase. El segundo algoritmo consiste en un ensamble de selectores, el cual considera una etapa de exclusión seguida por una de inclusión. Mientras el tercer algoritmo realiza una selección basada en agrupamiento. En lo que respecta a la selección de instancias en el espacio n-dimensional, tres algoritmos son propuestos, basados en las estrategias previamente presentadas en el espacio de los grafos. Por último, dos algoritmos híbridos son desarrollados, utilizando una estrategia de reducción en el espacio de los grafos y una de inclusión de instancias en el espacio n-dimensional.

Con base en los resultados experimentales obtenidos en el presente trabajo de investigación, los algoritmos propuestos en el espacio de los grafos producen una selección de grafos (instancias) que permite obtener los mejores resultados de clasificación de entre los diferentes enfoques de selección propuestos en esta tesis, obteniendo incluso una mejora estadísticamente significativa con respecto a trabajos de referencia utilizados en nuestra comparación.

Abstract

Graph supervised classification is an activity that has generated great interest in several application domains in science and engineering. This activity is related to building a model (classifier) that can be used to determine the class for a new graph, by means of the use of a graph training set. However, as n-dimensional data, there are some graphs in a training set which are not useful to classify, therefore those unnecessary elements from this set should be discarded. This process is known as instance selection and its main objective is to reduce the size of the training set, maintaining as much as possible, the same classification quality as the original training set. Instance selection algorithms for graph databases have not been reported in the literature and only this kind of algorithms have been developed for n-dimensional spaces.

In this research, we address the problem of instance selection for graph databases following three approaches. First, instance selection is carried out into the graph space, using a dissimilarity measure designed for comparing any two elements in the graph space. Second, instace selection is performed into the n-dimensional space generated by graph embedding via dissimilarity. Third, a hybrid instance selection is developed by combining both approaches.

In particular, into the graph space, three instance selection algorithms are proposed. The first algorithm processes each element in order of their average dissimilarity and discards those elements from the training set that do not affect the clasification quality. The second algorithm consist in an assembly of instance selection algorithms that considers an exclusion step followed by an inclusion step. The third algorithm performs a selection based on clustering. On the other hand, for instance selection into the n-dimensional space, three algorithms are proposed,

using the selection strategies of the first approach. Finally, for third approach two algorithms are developed using a reduction strategy into the graph space and an inclusion strategy into the n-dimensional space.

According to the experimental results obtained in this thesis, the algorithms proposed into the graph space produce an instance selection that can achieve the best classification results among the different selection approaches proposed in this thesis, obtaining even a significant statistically improvement over reference works used in our comparison.

Agradecimientos

Quiero agradecer primeramente a mis asesores, los doctores Ariel Carrasco Ochoa y Francisco Martínez Trinidad, por su paciencia, enseñanzas y consejos hacia mi persona durante el desarrollo de este trabajo de investigación. Me resulta gratificante aprender de dos buenos mentores, ¡Muchas gracias!.

Asimismo, agradezco a mi comité evaluador, los doctores Manuel Montes, Hugo Jair Escalante y Claudia Feregrino, por sus comentarios y sugerencias para seguir mejorando como profesional y como persona.

Muchas gracias a mis padres Bulmaro y Ana, porque en los buenos momentos y en los no tan buenos tengo su apoyo incondicional. De igual manera agradezco a Diego, porque, además de un buen hermano, tengo un buen amigo en el que puedo confiar. En general, gracias a los tres, por ser mi motivación más grande para seguir adelante.

Les doy las gracias a mis carnales que conocí en INAOE: Emmanuel Bolaños (Manolo), Alejandro Torres (Alex), Joel Rivas (hermanazo), Martín Letras (Marty), Nahum Sánchez (el doctor) y Luis Pellegrin (sensei), a los cuales aprecio y valoro por su amistad sincera, "muchas gracias de verdad". No olvido agradecer también a mis hermanos cubanos, Octavio y Niusvel, por compartirme sus conocimientos.

Por último, agradezco al Consejo Nacional de Ciencia y Tecnología por la beca asignada para realizar mis estudios de Maestría en el INAOE.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En la actualidad, ha surgido un creciente interés en utilizar grafos para el modelado de diversos tipos de objetos, debido a que los grafos permiten representar no sólo las características, sino también las relaciones existentes entre las diferentes partes de estos objetos (Riesen and Bunke, 2009c). Es posible encontrar a los grafos en diversas aplicaciones, tales como medicina molecular (Lozano and Escolano, 2006), redes de computadoras (Bunke, 2006), análisis de redes sociales (Chierichetti et al., 2009), reconocimiento de rostros (Han et al., 2012) y segmentación de imágenes (Harchaoui and Bach, 2007), entre otras.

En muchas de las aplicaciones mencionadas, una de las actividades más comunes es la clasificación supervisada de grafos. Esta actividad consiste en construir un modelo o clasificador que permita asignar la clase a la que pertenece un nuevo grafo, usando para ello un conjunto de grafos de entrenamiento previamente etiquetados (Riesen and Bunke, 2009c). Algunas aplicaciones que llevan a cabo clasificación supervisada de grafos son el diagnóstico médico (de Assis Zampirolli et al., 2010; Sharma et al., 2012), la clasificación de huellas dactilares (Marcialis et al., 2007), el reconocimiento de rostros (Han et al., 2012) o la categorización de documentos (Bunke and Riesen, 2011). En estas aplicaciones no todos los grafos que se emplean al clasificar son de utilidad, pues existen grafos redundantes que no aportan información relevante, así como grafos ruidosos (mal etiquetados) que provocan errores en la clasificación.

Ante los problemas expuestos previamente, en la literatura se ha propuesto

una línea de investigación denominada *selección de instancias* (García et al., 2015; Olvera-López et al., 2010). La idea de esta línea de investigación consiste en, dado un conjunto de entrenamiento, reducir el mismo de manera tal que el nuevo conjunto no contenga instancias ruidosas y/o redundantes, permitiendo mantener, tanto como sea posible, la misma calidad de clasificación que el conjunto original; consecuentemente los tiempos de ejecución en los procesos de clasificación y/o entrenamiento también se reducen.

A pesar del creciente interés en el uso de grafos para modelar diversos tipos objetos y de su uso en problemas de clasificación, en la literatura no se reportan algoritmos diseñados para el problema de la selección de instancias en bases de datos de grafos. Por lo que este trabajo de tesis se enfoca en proponer algoritmos para solucionar este problema, permitiendo obtener un conjunto reducido de grafos para entrenamiento sin afectar demasiado la calidad de clasificación.

1.1. Descripción del problema

Diversos enfoques han sido propuestos en la literatura para llevar a cabo tareas de clasificación sobre bases de datos de grafos, esto debido al creciente interés en diferentes dominios de aplicación por el uso de grafos para representar a los objetos bajo estudio (Bunke and Riesen, 2011; Harchaoui and Bach, 2007; Marcialis et al., 2007; Sharma et al., 2012). Sin embargo, en este tipo de aplicaciones al igual que ocurre con datos n-dimensionales, cuando el número de ejemplos en el conjunto de entrenamiento se incrementa, el tiempo que emplea el clasificador se ve afectado en las etapas de entrenamiento o clasificación.

Un problema muy común es que no todos los grafos en el conjunto de entrenamiento proporcionan información relevante para el proceso de clasificación, porque pueden encontrarse elementos redundantes o ruidosos. Consecuentemente, una selección de instancias resulta indispensable para descartar este tipo de elementos, de tal manera que su eliminación permita mantener, tanto como sea posible, la calidad de clasificación obtenida con el conjunto de entrenamiento original.

Con base en la revisión de la literatura, no se reportan algoritmos diseñados para el problema de selección de instancias en bases de datos de grafos, pues los algoritmos existentes han sido diseñados para representaciones n-dimensionales (García et al., 2015; Olvera-López et al., 2010).

1.2. Motivación

Abordar la problemática de selección de instancias en bases de datos de grafos, es una línea de investigación abierta y de mucho interés (Bunke and Riesen, 2011; de Assis Zampirolli et al., 2010; Sharma et al., 2012) para diversos dominios de aplicación, donde se requiere además de una buena clasificación, un tiempo de respuesta reducido. Por tal motivo, y ante la falta de algoritmos diseñados para este problema, el desarrollo de un algoritmo de selección de instancias para bases de datos de grafos toma una gran importancia.

La aplicación de un algoritmo de selección de instancias en un conjunto de entrenamiento permite obtener un conjunto reducido útil para la clasificación. El beneficio de utilizar este nuevo conjunto, es la reducción del tiempo empleado en los procesos de entrenamiento y clasificación, manteniendo una calidad de clasificación similar a la obtenida con el conjunto de entrenamiento original. Estos beneficios son la motivación principal para realizar la investigación descrita en la presente tesis, ya que no existen algoritmos de selección de instancias reportados en la literatura para trabajar con bases de grafos.

Con base en lo mencionado y considerando los aspectos del problema actual, en el presente trabajo de investigación se proponen diferentes soluciones al problema de selección de instancias en bases de datos de grafos.

1.3. Objetivos

De acuerdo a lo descrito previamente, los objetivos planteados en este trabajo de investigación son los siguientes:

1.3.1. Objetivo general

Desarrollar un algoritmo de selección de instancias para bases de datos de grafos, que permita reducir el tamaño del conjunto de entrenamiento, manteniendo, tanto como sea posible, los resultados de clasificación obtenidos con el conjunto original, y que supere los resultados alcanzados por algunos trabajos relacionados relevantes.

1.3.2. Objetivos específicos

1. Seleccionar una medida de disimilaridad para la comparación entre grafos que permita realizar la selección de instancias.
2. Proponer una estrategia de selección de instancias en el espacio de grafos que integre la medida de disimilaridad del paso anterior.
3. Proponer una estrategia de selección de instancias en el espacio n-dimensional generado por la aplicación de una técnica de embedding de grafos.
4. Desarrollar una estrategia de selección de instancias híbrida que combine las estrategias de los dos pasos anteriores.

1.4. Organización de esta tesis

El contenido de este trabajo está organizado de la siguiente manera:

En el capítulo 2 se presentan los conceptos básicos requeridos para entender el problema de selección de instancias en bases de datos de grafos; mismos que serán útiles para la comprensión del resto del documento.

En el capítulo 3 se describen los trabajos relacionados más relevantes que se han propuesto para esta línea de investigación. Se describen principalmente trabajos relacionados con la selección de instancias en espacios n-dimensionales, la selección de prototipos para embedding de grafos y la disimilaridad entre grafos.

En el capítulo 4 se presentan las soluciones propuestas en esta tesis para realizar selección de instancias en bases de datos de grafos.

El capítulo 5 muestra los resultados experimentales obtenidos al evaluar el desempeño de las soluciones propuestas y una comparación experimental contra otros algoritmos de referencia.

Finalmente, se exponen las conclusiones y algunas direcciones a seguir como trabajo futuro.

Capítulo 2

Marco teórico

En este capítulo se presentan los conceptos básicos necesarios para entender el problema abordado en esta tesis. Primero, en la sección 2.1 nos enfocamos en el tema de la clasificación supervisada. Posteriormente, en la sección 2.2, se explica la selección de instancias. En la sección 2.3, se presentan los conceptos y notación básica para la disimilaridad entre grafos. Por último, en la sección 2.4, se da una breve explicación del *embedding de grafos vía disimilaridad*, técnica usada para generar el espacio de representación n-dimensional.

2.1. Clasificación supervisada

La *clasificación supervisada*, en el reconocimiento de patrones, se refiere al proceso de determinar la clase o categoría a la que pertenece un nuevo objeto (figura 2.1), usando un conjunto de ejemplos previamente categorizados (*conjunto de entrenamiento*) (Bishop, 2006). Aplicaciones de clasificación supervisada pueden ser encontrados en diversos dominios de aplicación como el diagnóstico médico (Sharma et al., 2012), el reconocimiento de huellas dactilares (Marcialis et al., 2007), la clasificación de imágenes (Harchaoui and Bach, 2007), entre muchos otros.

El proceso de clasificación supervisada está conformado por dos etapas principales (Bishop, 2006): 1) una etapa de entrenamiento que sirve para inferir un modelo o conjunto de reglas para la clasificación, y 2) una etapa de clasificación de nuevos

objetos de los que se desconoce su *clase* o categoría. Una definición formal de clasificación supervisada es descrita a continuación:

Definición 1 (Clasificación supervisada)**.** Dado un conjunto de N objetos $T = \{(x_i, w_i)\}_{i=1,\ldots,N} \subseteq X \times C$, donde X es un dominio de objetos de cualquier tipo y $C = \{w_1, \ldots, w_k\}$ un conjunto finito de clases. La clasificación supervisada consiste en inferir, a partir de la información contenida en T, una función $f : X \to C$, tal que a cada objeto $x \in X$ le asigna una clase $w \in C$.

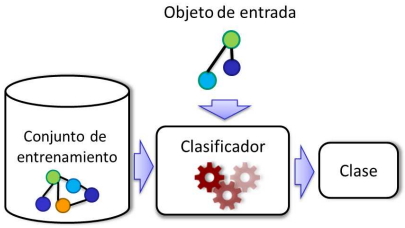

Figura 2.1: Proceso de la clasificación supervisada.

En la clasificación supervisada, una de las primeras preguntas es cómo podemos describir correctamente los objetos bajo estudio, de manera tal que un clasificador pueda ser aplicado. Para este propósito, es posible optar por dos enfoques:

- **El reconocimiento de patrones estadístico.** Los objetos son representados por vectores de n características, usualmente numéricas. Así, un objeto puede ser interpretado como un punto en el espacio real n-dimensional: $\mathbf{y} = (y_1, ..., y_n) \in \mathbb{R}^n$ (Bishop, 2006). La principal desventaja de este enfoque radica en la capacidad de representar sólo un número fijo de características y la falta de utilidad para representar las relaciones existentes entre las partes de un objeto.

- **El reconocimiento de patrones estructural.** Este enfoque representa a los objetos mediante estructuras de datos como árboles, cadenas o grafos (Conte et al., 2004). Dentro de estas estructuras, los grafos están entre los más utilizados, ya que permiten representar no sólo un número diferente de características, sino las relaciones existentes entre las diferentes partes de un objeto. De

ahí que diferentes dominios de aplicación están interesados en su uso, tales como la categorización de documentos (Bunke and Riesen, 2011), la segmentación de imágenes (Harchaoui and Bach, 2007) o el reconocimiento de rostros (Han et al., 2012).

Por lo anterior, y debido a que el problema a resolver en el presente trabajo de investigación gira entorno a la clasificación supervisada de grafos, especialmente sobre grafos etiquetados, damos una definición formal de éstos:

Definición 2 (Grafo etiquetado)**.** Sean L_V y L_E conjuntos de etiquetas de vértices y aristas respectivamente. Un *grafo etiquetado* es una 4-tupla, $g = (V, E, \mu, \nu)$, donde:

- V es un conjunto finito de vértices.
- $E \subseteq V \times V$ es el conjunto de aristas.
- $\mu : V \to L_V$ es una *función etiquetadora* encargada de asignar etiquetas a los vértices.
- $\nu : E \to L_E$ es una *función etiquetadora* encargada de asignar etiquetas a las aristas.

Por otra parte, en el contexto de esta tesis una *instancia* es un objeto de un dominio particular, el cual puede ser representado como un grafo o un vector n-dimensional de características.

2.1.1. Clasificador k-Vecinos más cercanos

En la literatura es posible encontrar diferentes enfoques para la clasificación de grafos (Bunke and Riesen, 2011; Han et al., 2012), siendo uno de los enfoques más comunes el de k-Vecinos más cercanos (k-NN por sus siglas en inglés *k-Nearest Neighbor*). En la presente tesis, k-NN fue seleccionado para llevar a cabo diferentes pruebas experimentales debido a su simplicidad y al hecho de que la mayoría de los selectores de instancias para espacios n-dimensionales han sido diseñados para este clasificador. Además, este clasificador puede ser usado directamente en el espacio de los grafos, mediante una función de similitud definida en este espacio. Por este

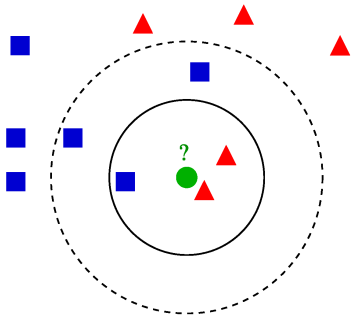

Figura 2.2: Clasificación de un objeto por el clasificador k-NN.

motivo, en el resto de esta sección damos una breve explicación de su funcionamiento.

k-NN (Cover and Hart, 1967) es un tipo de clasificador no paramétrico, el cual no construye un modelo para realizar clasificación. Así, todas las instancias en el conjunto de entrenamiento son utilizadas cada vez que se requiere clasificar un nuevo objeto.

La idea del clasificador k-NN consiste en calcular las distancias entre una instancia g y las N instancias de ejemplo en el conjunto de entrenamiento T. Con base en las distancias calculadas, la clase asignada a g está en función de la clase más frecuente de las k instancias en T más cercanas a g.

Un ejemplo del algoritmo k-NN es mostrado en la figura 2.2. En este ejemplo se debe determinar la clase a la que pertenece un objeto círculo. Con base en la regla de la clase más frecuente para sus tres vecinos más cercanos, el objeto círculo es clasificado como un triángulo.

2.2. Selección de instancias

Uno de los principales factores que afectan el rendimiento de un clasificador es el incremento en el número de ejemplos en el conjunto de entrenamiento (García et al., 2015). Además, algunos elementos dentro de ese conjunto no son de gran utilidad para el proceso de entrenamiento y clasificación. Este tipo de elementos pueden ser: 1) elementos ruidosos, que son aquellos ejemplos mal etiquetados y que pueden provocar una clasificación errónea, o 2) elementos redundantes, que no proporcionan información adicional y que son poco necesarios para el conjunto de entrenamiento.

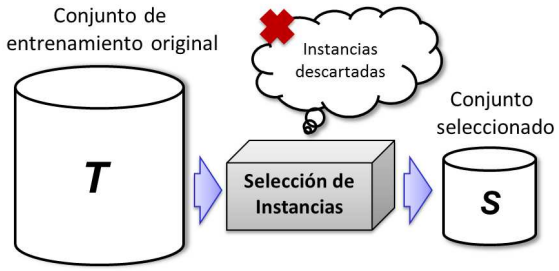

Figura 2.3: Proceso de selección de instancias.

Con base en lo antes mencionado, es necesario un proceso de selección de elementos del conjunto de entrenamiento que sean útiles para la clasificación (ver figura 2.3). Este proceso es conocido como selección de instancias (García et al., 2015; Olvera-López et al., 2010) y es el problema a resolver en este trabajo de investigación.

Definición 3 (Selección de instancias)**.** Dado un conjunto de entrenamiento T, el proceso de selección de instancias consiste en elegir un subconjunto $S \subset T$, de tal manera que S no contenga elementos ruidosos y/o redundantes, y S permita mantener, tanto como sea posible, la misma calidad de clasificación que T.

Como el conjunto de entrenamiento es reducido mediante la selección de instancias, los tiempos de ejecución de los procesos de entrenamiento y clasificación también se reducen.

De acuerdo a Olvera-López et al. (2010), existen dos tipos de estrategias para llevar a cabo la selección de instancias en la clasificación supervisada:

- **Wrapper:** El criterio de selección de las instancias está basado en los resultados obtenidos por un clasificador específico. En la literatura, una gran cantidad de trabajos para representaciones n-dimensionales se han propuesto con base en la regla del vecino más cercano o bien en su regla más general k-NN. La idea de la estrategia tipo *wrapper* consiste en descartar aquellas instancias que no contribuyen con la precisión de la clasificación de acuerdo a la evaluación obtenida por un clasificador (k-NN por ejemplo). El utilizar un clasificador

para la selección de instancias permite obtener una buena estimación para este clasificador. Sin embargo, los resultados pueden no ser buenos si se usa un clasificador diferente.

- **Filter:** El criterio de selección de las instancias se basa en una función independiente de algún clasificador. Un procedimiento muy utilizado en estrategias tipo $filter$ consiste en identificar las instancias ubicadas en la frontera de decisión y aquellas que se encuentran en el interior de una clase. Las instancias frontera son aquellas instancias más cercanas a instancias de clase diferente, mientras que las instancias ubicadas al interior de la clase son aquellas que no tienen esta característica. Algunos algoritmos de tipo $filter$ se enfocan en seleccionar instancias frontera, pues éstas proporcionan información usual para preservar una frontera de decisión entre clases.

En este trabajo de investigación exploramos ambos tipos de estrategias, por lo que en el siguiente capítulo se explican con detalle algunos de los algoritmos de tipo $wrapper$ y $filter$ más exitosos reportados en la literatura, todos ellos diseñados para representaciones n-dimensionales.

2.3. Disimilaridad entre grafos

El problema de la disimilaridad entre grafos toma una gran importancia para el desarrollo de esta tesis, pues con base en los objetivos planteados en la sección 1.3, una medida de disimilaridad nos permitirá proponer una estrategia de selección de instancias que trabaje directamente en el espacio de los grafos.

La disimilaridad entre grafos está relacionada con la *correspondencia de grafos* (Bunke, 2000; Conte et al., 2004), cuyo propósito es encontrar las correspondencias entre las subestructuras de dos grafos a comparar, satisfaciendo en mayor o menor medida ciertas condiciones. Basados en este concepto, la disimilaridad de acuerdo a Borgwardt et al. (2007) puede ser definida como:

Definición 4 (Disimilaridad entre grafos)**.** Dado dos grafos g_1 y g_2 de un dominio de grafos G, el problema de la disimilaridad entre grafos consiste en encontrar una función:

$$d : G \times G \longrightarrow \mathbb{R}$$

tal que $d(g_1, g_2)$ cuantifica la disimilaridad entre g_1 y g_2.

Existen dos enfoques para llevar a cabo la correspondencia de grafos (Bunke, 2000; Conte et al., 2004; Riesen, 2009): *la exacta y la inexacta*. El propósito de la *correspondencia exacta* consiste en determinar si dos grafos, o al menos parte de ellos, son idénticos en términos de estructura y etiquetas. No obstante, debido a la variabilidad de los objetos bajo estudio y errores en el proceso de extracción de los grafos, es difícil encontrar dos grafos de la misma clase con una estructura idéntica o al menos una gran parte de ella. Por otro lado, la *correspondencia inexacta* (o *correspondencia con tolerancia a errores*) permite superar estas limitantes, pues hace posible el mapeo entre subestructuras de dos grafos incluso si las estructuras y etiquetas asignadas en los vértices o aristas no son idénticas. Para esto, un costo en el mapeo es asignado, tal que valores altos indican que dos grafos son disimilares y valores bajos lo contrario.

Aunque varios enfoques de correspondencia inexacta han sido propuestos en la literatura (ver sección 3.3), uno de los más conocidos es la *distancia de edición de grafos*, de la cual se da una descripción a detalle en la siguiente sección.

2.3.1. Distancia de Edición de Grafos

La idea de la *distancia de edición de grafos* (GED por sus siglas en inglés *Graph Edit Distance*), es evaluar la disimilaridad entre dos grafos mediante la mínima cantidad de operaciones de edición requeridas para transformar un grafo g_1 en un grafo g_2 (Conte et al., 2004; Riesen, 2009; Riesen and Bunke, 2009a). Cada operación de edición e_i consiste en la *inserción*, *eliminación* o *sustitución* de un vértice o arista. En la literatura (Riesen and Bunke, 2009a) la sustitución de un vértice u por v es denotada por $(u \rightarrow v)$, la eliminación de un vértice u por $(u \rightarrow \varepsilon)$ y la inserción de un vértice v por $(\varepsilon \rightarrow v)$. La notación utilizada para el caso de las aristas es muy similar.

Con base en lo anterior, un grafo g_1 puede ser transformado en un grafo g_2 aplicando una secuencia de operaciones de edición $(e_1, ..., e_k)$. Esta secuencia de operaciones comúnmente es llamada camino de edición y un ejemplo es mostrado en la figura 2.4. En este ejemplo, las aristas adyacentes al vértice D son eliminadas

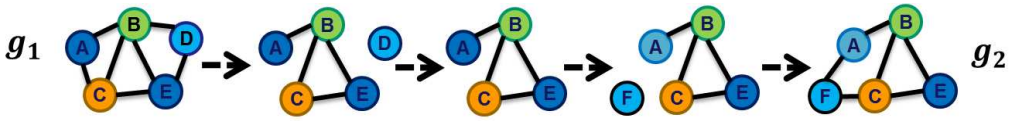

Figura 2.4: Un posible camino de edición entre un par de grafos g_1 y g_2.

del grafo g_1, para luego eliminar el vértice D; posteriormente, un nuevo vértice F es añadido, así como las aristas que lo conectan con los vértices A y C. Cabe señalar que para cada par de grafos pueden existir varios caminos de edición diferentes, lo cual depende del número de vértices y aristas de los grafos a comparar. De acuerdo a Riesen and Bunke (2009a), al conjunto de todos los posibles caminos de edición se le denota como $E(g_1, g_2)$. Así, para el ejemplo de la figura 2.4, el vértice D puede ser sustituido por F, en lugar de eliminar D y luego insertar un nuevo vértice F.

Para encontrar el mejor camino de edición de entre todos los caminos $E(g_1, g_2)$, a cada operación de edición e_i le es asociada una función de costo. Esta función, denotada por $c(e_i)$, es utilizada para determinar el costo de cada operación de edición e_i en un camino de edición. Con esto, la distancia de edición de dos grafos se define como el camino de edición de costo mínimo (Gao et al., 2010; Riesen and Bunke, 2009a). Valores bajos de la distancia de edición indican que dos grafos son muy similares, mientras que valores altos indican lo contrario.

Definición 5 (Distancia de edición de Grafos)**.** Dados los grafos $g_1 = (V_1, E_1, \mu_1, \nu_1)$ y $g_2 = (V_2, E_2, \mu_2, \nu_2)$, la distancia de edición entre g_1 y g_2 está definida como:

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in E(g_1,g_2)} \sum_{i=1}^{k} c(e_i), \tag{2.3.1}$$

donde $E(g_1, g_2)$ denota el conjunto de caminos de edición entre g_1 y g_2, $c(e_i)$ es la función de costo y e_i es la i-ésima operación de edición de un camino.

Ejemplo de una función de costo

La elección de una función de costo adecuada es una tarea importante para calcular la distancia de edición entre un par de grafos, pues ésta determina el costo o importancia de cada tipo de operación de edición. Una función de costo muy utilizada

en varios dominios de aplicación es la basada en la distancia Euclidiana (Riesen, 2009).

Definición 6 (Función de costo Euclidiana)**.** Dado los grafos $g_1 = (V_1, E_1, \mu_1, \nu_1)$ y $g_2 = (V_2, E_2, \mu_2, \nu_2)$, con $L_N, L_E \subseteq \mathbb{R}^n$. La función de costo euclidiana para los vértices $u \in V_1$, $v \in V_2$ y aristas $p \in E_1$, $q \in E_2$ consiste en:

$$
\begin{aligned}
c(u \to \varepsilon) &= \alpha \cdot \tau_{vertice} \\
c(\varepsilon \to v) &= \alpha \cdot \tau_{vertice} \\
c(u \to v) &= \alpha \cdot d_E(\mu_1(u) - \mu_2(v)) \\
c(p \to \varepsilon) &= (1 - \alpha) \cdot \tau_{arista} \\
c(\varepsilon \to q) &= (1 - \alpha) \cdot \tau_{arista} \\
c(p \to q) &= (1 - \alpha) \cdot d_E(\nu_1(p) - \nu_2(q))
\end{aligned}
$$

donde $\tau_{vertice}$, $\tau_{arista} \in \mathbb{R}^+$ son valores constantes que representan el costo de la inserción y/o eliminación de vértices y aristas, respectivamente; mientras que $\alpha \in [0, 1]$ es un parámetro de control que indica la importancia relativa entre las operaciones de edición de los vértices o de las aristas. Por último, d_E es la distancia Euclidiana en $\mathbb{R}^n$, la cual determina el costo por sustituir el vértice u por v, usando la información de las etiquetas $\mu_1(u)$ y $\mu_2(v)$ (valores reales para este ejemplo en específico) asociadas a estos vértices.

Cabe señalar que para un dominio en particular no existe una única función de costo (Riesen, 2009). Por lo que otras funciones de costo pueden ser propuestas basándose en el conocimiento previo de las etiquetas asociadas a los vértices y aristas de los grafos a comparar.

2.4. Embedding de grafos vía disimilaridad

En esta sección se da una breve explicación de los conceptos relacionados con el *embedding de grafos vía disimilaridad*, pues de acuerdo a los objetivos específicos 3 y 4 de la sección 1.3, esta técnica será de utilidad para generar un espacio de representación n-dimensional; en dicho espacio, los algoritmos de selección de instancias diseñados para espacios n-dimensionales pueden ser aplicados.

En la actualidad el *embedding de grafos* ha tomado un gran interés dentro del área de reconocimiento de patrones estructural, especialmente en técnicas basadas en la representación por disimilaridades. La idea de este *embedding de grafos* radica en transformar un grafo en un vector de características, usando un conjunto de grafos prototipo y una medida de disimilaridad (Riesen and Bunke, 2009c). Este tipo de técnicas surgen como un enfoque para superar la falta de herrramientas algorítmicas en el dominio de los grafos y aprovechar las técnicas existentes en el reconocimiento de patrones estadístico (Bunke and Riesen, 2008).

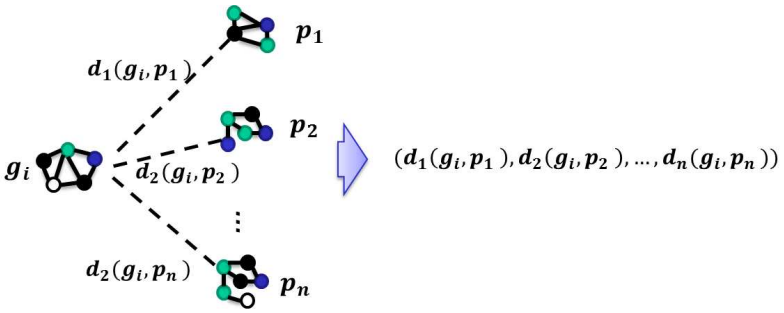

Figura 2.5: Un grafo g_i es representado como un vector de características de n dimensiones $(d(g_i, p_1), ..., d(g_i, p_n))$.

Dado un conjunto de grafos $T = \{g_1, \ldots, g_m\}$, de algún dominio de grafos G, y una medida de disimilaridad entre grafos $d(g_i, g_j)$. La selección de prototipos para *embedding de grafos vía disimilaridad* consiste en seleccionar un conjunto de grafos prototipo $P = \{p_1, \ldots, p_n\} \subseteq T$ con $n \leq m$, y calcular la medida de disimilaridad entre cada grafo de entrada g_i y los n grafos prototipo en P (ver figura 2.5). Estas n disimilaridades son usadas para construir un vector de características de n dimensiones $(d(g_i, p_1), ..., d(g_i, p_n))$, el cual representa al grafo g_i. Basados en esta técnica, un conjunto de grafos puede ser transformado en un conjunto de vectores n-dimensionales de características.

Definición 7 (Embedding de grafos vía disimilaridad)**.** Dado un dominio de grafos G, una función de disimilaridad $d : G \times G \longrightarrow \mathbb{R}$ y un conjunto de grafos prototipo $P = \{p_1, ..., p_n\} \subseteq G$. El mapeo $\varphi_n^P : G \to \mathbb{R}^n$ está definido como:

$$\varphi_n^P(g) = (d(g, p_1), ..., d(g, p_n))$$

Un estudio de los trabajos más relevantes relacionados con esta técnica es descrito en el siguiente capítulo.

Capítulo 3

Trabajo relacionado

En el presente capítulo se describen los trabajos más relevantes que se han reportado en la literatura relacionados con el problema abordado en esta tesis. Primero, en la sección 3.1 se estudian los algoritmos de selección de instancias para espacios n-dimensionales. Posteriormente, en la sección 3.2 se analizan los algoritmos de selección de prototipos para embedding de grafos, los cuales reducen la dimensionalidad del espacio de representación alternativo al de los grafos, pero no el conjunto de entrenamiento. Por último, en la sección 3.3 se revisan los diferentes enfoques para determinar la disimilaridad entre grafos, haciendo énfasis en la distancia de edición.

3.1. Selección de instancias para representaciones n-dimensionales

Como fue mencionado en la sección 2.1, en el reconocimiento de patrones estadístico los objetos son representados por vectores n-dimensionales de características, usualmente numéricas. De esta manera, un objeto puede ser interpretado como un punto en el espacio real n-dimensional $\mathbb{R}^n$ (Bishop, 2006). En la literatura, los algoritmos que solucionan el problema de la selección de instancias están diseñados para este tipo de representaciones. Estos algoritmos constituyen una opción para resolver el problema planteado a esta tesis, pues a partir de bases de datos de grafos es posible generar un espacio real n-dimensional mediante el *embedding de grafos*

via disimilaridad. Por este motivo, en esta sección se presenta una revisión de los algoritmos de selección de instancias más exitosos reportados en la literatura para espacios n-dimensionales.

Con la finalidad de una mejor organización, los algoritmos descritos son divididos en dos grupos (Olvera-López et al., 2010): *wrapper* y *filter*. En el primer grupo se ubican aquellos algoritmos que sí utilizan un clasificador en específico como criterio de selección de instancias, mientras que en el segundo grupo son ubicados aquellos que no utilizan un clasificador como criterio de selección.

3.1.1. Algoritmos wrapper

Los algoritmos de selección de instancias DROP 1-5 (*Decremental Reduction Optimization Procedure*) (Wilson and Martinez, 2000) están basados en el concepto de socios de un instancia en el conjunto de entrenamiento. Los socios de una instancia p, son aquellas instancias que tiene a p como vecina más cercana. DROP1 elimina una instancia p del conjunto de entrenamiento si los socios de p se clasifican correctamente sin p; la calidad de clasificación es evaluada sobre el mismo subconjunto seleccionado S. El funcionamiento de DROP2 es similar a DROP1 con la diferencia que S es evaluado sobre el conjunto de entrenamiento T en lugar de S. DROP2 cambia el orden en el que las instancias son procesadas, iniciando por aquellas instancias que se encuentran lo más lejos posible de la frontera de decisión. Los algoritmos DROP3, DROP4 y DROP5 utilizan primero un filtro de ruido con el propósito de eliminar todas las posibles instancias ruidosas en el conjunto de entrenamiento. La diferencia entre estos últimos algoritmos radica en el filtro utilizado, posteriormente su funcionamiento es igual al de DROP2.

En (Brighton and Mellish, 2002) se propone *Iterative Case Filtering* (ICF), cuya selección está basada en los conceptos de $Reacheable(p)$ y $Coverage(p)$, que son los conjuntos de instancias vecinas más cercanas y socios respectivamente. La regla de ICF consiste en eliminar aquellas instancias en las que el valor de $Reachable(p)$ es mayor al de $Coverage(p)$. En otras palabras, una instancia p es eliminada si otras instancias proveen la misma información que p. En una etapa inicial ICF utiliza un filtro de ruido mediante el algoritmo *Edited Nearest Neighbor* (ENN) (Wilson, 1972).

El algoritmo CPruner presentado en (Zhao et al., 2003) se basa también en los conceptos $Reacheable(p)$ y $Coverage(p)$, procurando que los socios de una instancia p pertenezcan a la misma clase. Este algoritmo elimina las instancias que son redundantes y ruidosas, tratando de preservar instancias críticas. Una instancia es crítica cuando su eliminación afecta la clasificación de otras instancias. Por otra parte, una instancia p es denominada ruidosa cuando el valor $Reachable(p)$ es mayor al de $Coverage(p)$, mientras una instancia p es redundante cuando ésta es clasificada correctamente por $Reachable(p)$. El algoritmo CPruner toma en cuenta el orden en el que son descartadas las instancias, el cual está basado en el número de instancias vecinas y enemigas (instancias vecinas de clase diferente) más cercanas.

El algoritmo *Generalized Condensed Nearest Neighbor* (GCNN) (Chou et al., 2006) es una mejora al algoritmo CNN (Hart, 1968). El subconjunto a seleccionar S se inicia con una instancia por cada clase. Luego, si cada instancia del conjunto de entrenamiento T es mal clasificada, entonces es incluida en S. Una instancia p es clasificada correctamente sólo si su instancia vecina más cercana q en S es de su misma clase, y la distancia entre p y q es menor que la distancia entre p y su instancia enemiga más cercana en S. Este proceso termina hasta que todas las instancias en el conjunto de entrenamiento son clasificadas correctamente.

En (Marchiori, 2010) se propone el algoritmo *Class Conditional Instance Selection* (CCIS), el cual consta de dos etapas para la selección de instancias. En la primera etapa, CCIS construye dos grafos denominados de proximidad: el primero de ellos representa la relación entre cada instancia p del conjunto de entrenamiento T y las instancias vecinas más cercanas de su misma clase; mientras que el segundo grafo representa la relación entre p y sus instancias enemigas más cercanas. Estos grafos son usados para definir una función de puntuación basada en la teoría de la información dirigida, la cual es aplicada sobre los vértices de cada grafo (un vértice representa a una instancia p) para medir su valor de distribución dentro de estos grafos. A partir de esta función de puntuación, CCIS determina el puntaje para todas las instancias en T y analiza iterativamente cada una de ellas, seleccionando las instancias con un puntaje alto. Este proceso continúa mientras el error en la calidad de clasificación sobre T disminuye. Con la finalidad de obtener una mejor reducción del conjunto

seleccionado, un algoritmo denominado *Thin* es aplicado sobre éste en una segunda etapa. *Thin* descarta del conjunto de instancias seleccionadas durante la primera etapa aquellas que al ser descartadas no afectan negativamente la calidad de clasificación.

El algoritmo *Instance Selection Based on Ranking* (ISR) (Vallejo et al., 2010) calcula la relevancia de cada instancia basándose en el promedio de disimilaridad de una instancia con las instancias en su clase y el promedio de disimilaridad con las instancias de clase diferente. Luego, ISR selecciona una a una las instancias (en orden de relevancia) hasta clasificar correctamente las instancias en el conjunto de entrenamiento.

3.1.2. Algoritmos filter

El algoritmo *Pairwise Opossite Class-Nearest Neighbor* (POC-NN) propuesto por Raicharoen and Lursinsap (2005), selecciona instancias cercanas a las regiones de la frontera de decisión de cada clase. Para esto, POC-NN calcula la media de las instancias en cada clase. Posteriormente, para encontrar una instancia frontera p_f en la clase C_1, este algoritmo calcula la media m_2 de las instancias en la clase opuesta C_2. Así, la instancia frontera p_f es la instancia de la clase C_1 más cercana a m_2. Las instancias frontera en la clase C_2 son encontradas de manera similar. Finalmente, POC-NN selecciona sólo las instancias ubicadas en la frontera de decisión de cada clase.

Prototype Selection by Relevance (PSR) propuesto por Olvera-López et al. (2008), calcula la relevancia de todas las instancias en el conjunto de entrenamiento como el promedio de disimilaridad de una instancia con las instancias en su clase. Las instancias son ordenadas con base en su relevancia y un porcentaje de éstas es seleccionado. Luego, para cada instancia seleccionada se calcula su instancia más similar de clase diferente (instancia frontera). PSR selecciona como conjunto final un porcentaje de instancias relevantes y las instancias frontera calculadas a partir de éstas.

Prototype Selection by Clustering (PSC) (Olvera-López et al., 2010) selecciona instancias ubicadas en la frontera de decisión de cada clase y algunas ubicadas en

el interior. PSC divide el conjunto de entrenamiento en C grupos y analiza cada uno de éstos para determinar si son homogéneos o no. Si un grupo es homogéneo (contiene instancias de la misma clase), PSC retiene el centroide y descarta el resto de instancias en el grupo. En caso contrario, cuando un grupo es no homogéneo (contiene instancias de dos o más clases) selecciona las instancias frontera. El conjunto final seleccionado por PSC está conformado por los centros de cada grupo y las instancias ubicadas en la frontera de decisión de cada clase.

El algoritmo *Class Boundary Preserving* (CBP) (Nikolaidis et al., 2011) aplica primero el filtro ENN (Wilson, 1972) para descartar instancias ruidosas. Posteriormente, CBP realiza una distinción entre las instancias frontera de las no frontera, calculando para cada instancia p en el conjunto de entrenamiento los conjuntos de instancias $R(p)$ y $E(p)$, que son las instancias que clasifican a p correctamente con 1-NN y las instancias enemigas más cercanas. CBP utiliza la función de disimilaridad *Coseno* para calcular el valor de cercanía entre cada instancia p y las instancias dentro de $R(p)$ y $E(p)$. Valores negativos de esta función indican que p está ubicada en la frontera de decisión. Así, CBP separa las instancias frontera de las no frontera, comparando los valores obtenidos por la función *Coseno* para cada instancia con respecto a un umbral. Posteriormente, este algoritmo determina los pares de instancias enemigas mutuas (par de instancias más cercanas entre sí de clase diferente) de entre las instancias frontera identificadas en el paso anterior. Por último, un algoritmo de agrupamiento es aplicado sobre las instancias no frontera y los centros de cada grupo son seleccionados. El subconjunto final seleccionado por CBP está conformado por instancias enemigas mutuas (ubicadas en la frontera de decisión) y los centros de cada grupo.

En (Hernandez-Leal et al., 2013) se propone *InstanceRank based on Borders* (IRB), el cual utiliza primero el filtro de ruido ENN (Wilson, 1972) para suavizar la frontera de decisión de cada clase. Después, IRB calcula la relevancia de cada instancia con base en sus k enemigos más cercanos. Las instancias son ordenadas basadas en esta relevancia y un porcentaje de ellas es seleccionado.

En esta sección se revisaron algunos de los algoritmos de selección de instancias más exitosos reportados en la literatura. Primero se describen los algoritmos de tipo

wrapper, que si bien son lentos, obtienen los mejores resultados de clasificación. Luego, se presentan los algoritmos de tipo *filter*, los cuales son más rápidos; sin embargo, su principal desventaja es la tendencia a seleccionar una gran cantidad de instancias.

3.2. Selección de prototipos para embedding de grafos vía disimilaridad

En esta sección se describen los principales algoritmos propuestos en la literatura para la línea de investigación conocida como *selección de prototipos para embedding de grafos vía disimilaridad*. El principal propósito de esta línea de investigación consiste en generar un espacio de representación n-dimensional basado en disimilaridades, con la finalidad de aprovechar las ventajas de las diferentes técnicas y herramientas desarrolladas en el reconocimiento de patrones estadístico.

En (Pekalska et al., 2006) los autores proponen por primera vez el espacio de representación por disimilaridades para el reconocimiento de patrones. Esta técnica consiste en caracterizar los elementos de un conjunto de datos de entrada $S \subset X$ (donde X puede ser cualquier espacio de representación) en una matriz de disimilaridades $\mathbf{D}$, usando una medida de disimilaridad $d : X \times X \to \mathbb{R}$ y un conjunto de prototipos $P \subseteq S$. Esta idea fue extendida para grafos por Riesen et al. (2007) y es conocida como *embedding de grafos vía disimilaridad*.

Dado un conjunto de grafos $T = \{g_1, \ldots, g_m\}$ de algún dominio G, una medida de disimilaridad entre grafos $d(g_j, g_k)$ y un grafo de entrada g_i. La selección de prototipos para *embedding de grafos vía disimilaridad* selecciona un conjunto de grafos prototipo $P = \{p_1, \ldots, p_n\} \subseteq T$ con $n \leq m$, y calcula la disimilaridad entre g_i y los n prototipos en P. Estas n disimilaridades son usadas para construir un vector n-dimensional de características $(d(g_i, p_1), ..., d(g_i, p_n))$, el cual representa a g_i. Con esta técnica, un conjunto de grafos puede ser transformado en un conjunto de vectores de n-dimensiones. Sin embargo, en esta técnica un conjunto de prototipos pequeño es deseable, ya que esto permite generar un espacio de representación con una dimensionalidad reducida.

En la literatura diferentes algoritmos han sido propuestos para la *selección de prototipos para embedding de grafos vía disimilaridad*. En (Riesen and Bunke, 2009c; Riesen et al., 2007) por ejemplo, los autores proponen seis diferentes algoritmos, la idea principal de éstos es evitar la redundancia y seleccionar aquellos que incluyan la mayor información descriptiva de un conjunto de grafos. La ejecución de los algoritmos propuestos es llevada a cabo bajo dos enfoques: 1) la selección es realizada sobre todo el conjunto de grafos sin importar la clase a la que pertenece cada grafo seleccionado, y 2) la selección es ejecutada separadamente para cada una de las N clases del conjunto de entrenamiento. El segundo enfoque muestra un mejor rendimiento en la calidad de clasificación que el primero.

En (Riesen and Bunke, 2009b) se proponen varios algoritmos de reducción que usan el clasificador de vecinos más cercanos para determinar el conjunto de grafos prototipo. Estos esquemas de reducción eliminan la redundancia de grafos similares que pertenecen a la misma clase y seleccionan grafos significativos para la clasificación. Los esquemas propuestos se caracterizan por inferir de manera automática el número de prototipos.

En (Borzeshi et al., 2013) por otro lado, los autores proponen una extensión a los algoritmos de selección de prototipos propuestos en (Riesen and Bunke, 2009c; Riesen et al., 2007). La idea principal de su propuesta radica en considerar la discriminación entre clases. Para ello el conjunto de prototipos es elegido utilizando un peso asignado a la compacidad intra-clase y separación inter-clase. Los algoritmos presentados en (Borzeshi et al., 2013) demuestran obtener mejores resultados de clasificación que los algoritmos reportados en (Riesen and Bunke, 2009c; Riesen et al., 2007).

Un algoritmo denominado *Optimized Dissimilarity Space Embedding* (ODSE) es propuesto por Livi et al. (2014). La idea de ODSE se basa en el uso de un algoritmo genético bajo dos etapas principales: compresión y expansión. En la etapa de compresión se descartan los prototipos ubicados en el interior de la clase, mientras en la etapa de expansión se intenta seleccionar los prototipos ubicados en la frontera de decisión. Estas estrategias son aplicadas en el espacio de disimilaridad, posteriormente un mapeo es realizado en el espacio de los grafos para identificar

los prototipos seleccionados. Este procedimiento es repetido hasta encontrar un conjunto de prototipos que proporcione la mayor información descriptiva posible del conjunto original. Aunque el algoritmo optimiza de manera automática el número de prototipos, una de sus principales desventajas es su costo en el tiempo de ejecución.

Como puede verse, los algoritmos de *selección de prototipos para embedding de grafos* tienen como fin generar el espacio de espacio de representación n-dimensional con una dimensionalidad reducida. Sin embargo, todos los elementos del conjunto de entrenamiento en esta nueva representación son utilizados para entrenar al clasificador, es decir, estos algoritmos no reducen el conjunto de entrenamiento. En contraste, la *selección de instancias* tiene como finalidad obtener un conjunto de entrenamiento reducido, que mantenga, tanto como sea posible, la misma calidad de clasificación que el conjunto de entrenamiento original.

3.3. Disimilaridad entre grafos

En esta sección se presentan los enfoques relacionados con la disimilaridad entre grafos (ver figura 3.1) y que se serán de utilidad para el desarrollo de la presente tesis. Para esto, son estudiados dos enfoques: la *correspondencia de grafos exacta e inexacta*.

En el enfoque basado en la *correspondencia exacta*, se asume que es posible encontrar un isomorfismo entre grafos y subgrafos (Conte et al., 2004; Garey and Johnson, 1990). En el primer caso se determina si dos grafos son idénticos en términos de la estructura y las etiquetas de los grafos, mientras en el segundo caso se busca si un grafo está contenido de manera idéntica en otro de mayor tamaño. Por otro lado, métodos como el máximo subgrafo común (Bunke et al., 2002) y el mínimo supergrafo común (Bunke, 2000), son útiles para determinar si un par de grafos comparten subestructuras idénticas. A partir de estos dos últimos métodos una gran cantidad de medidas de disimilaridad se han propuesto en (Fernández and Valiente, 2001; Wallis et al., 2001).

El enfoque basado en *correspondencia inexacta (o con tolerancia a errores)* por otra parte, permite el mapeo entre subestructuras de dos grafos cuando las etiquetas de estas subestructuras no son completamente idénticas (ver sección 2.3) o incluso si

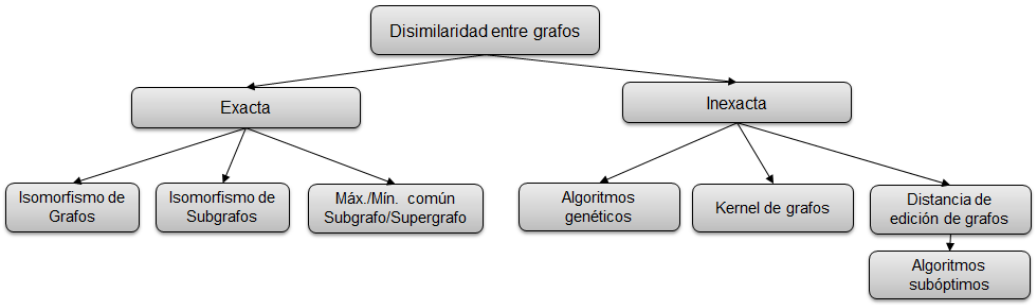

Figura 3.1: Enfoques para determinar la disimilaridad entre grafos.

las subestructuras no coinciden. En este enfoque algunos trabajos están basados en algoritmos genéticos (Auwatanamongkol, 2007; Cross et al., 1997; Suganthan, 2002), en los cuales los autores plantean el cálculo de la disimilaridad entre grafos como un problema de optimización combinacional. Los cromosomas representan las posibles correspondencias entre un par de grafos y diferentes operaciones de cruza o mutación son aplicados sobre estas correspondencias. En cada generación las correspondencias son ajustadas convenientemente atendiendo a una función *fitness*. El principal problema de estos trabajos radica en que sus resultados dependen en gran medida de los cromosomas iniciales. Además que, estos algoritmos son no deterministas, por lo tanto no siempre encuentran la misma solución.

Otros trabajos para determinar la disimilaridad entre grafos son los *kernels de grafos* (Bunke and Riesen, 2011; Gärtner et al., 2003; Harchaoui and Bach, 2007). La idea de los *kernels* consiste en el mapeo implícito de los dos grafos a comparar a una representación por medio de vectores de n dimensiones, posteriormente la disimilaridad es calculada como el producto escalar entre estos vectores. Métodos como los *kernels de convolución*, descomponen estructuras complejas en subestructuras más simples (Borgwardt et al., 2007; Ramon and Gärtner, 2003); no obstante, estos métodos son inaplicables en términos prácticos debido a su complejidad computacional, la cual es exponencial. Otros trabajos en esta dirección son los *kernels* basados en caminatas aleatorias (Borgwardt et al., 2005), los cuales proponen una solución con una complejidad polinomial; sin embargo, su funcionamiento está restringido para un determinado tipo de grafos. En general, el principal problema de los *kernels de grafos* radica en el diseño de un *kernel* adecuado, que describa las relaciones estructurales de los grafos a comparar, con un desempeño eficiente.

La distancia de edición (GED por sus siglas en inglés *Graph Edit Distance*) es uno de los enfoques más flexibles y versátiles para medir la disimilaridad usando correspondencia inexacta. Este enfoque determina la disimilaridad para cualquier tipo de grafos: etiquetados o no etiquetados, así como dirigidos o no dirigidos. Métodos en (Berretti et al., 2001; Gregory and Kittler, 2002), exploran todas las posibles transformaciones válidas entre dos grafos. Aunque estos métodos encuentran siempre una solución, si es que existe, su complejidad es exponencial con respecto al número de vértices entre los grafos a comparar. Otros métodos, como los presentados por Gao et al. (2010), están basados en redes neuronales, teoría de probabilidad y en el máximo subgrafo común o mínimo supergrafo común. Los primeros dos métodos sólo son funcionales para un determinado dominio de aplicación, mientras los basados en el máximo subgrafo común y mínimo supergrafo común no son útiles en aplicaciones reales debido a su complejidad exponencial.

En (Justice and Hero, 2006) por otra parte, se propone un método de programación lineal basado en el problema de asignación (Munkres, 1957). Este problema de asignación consiste en la tarea de encontrar una asignación óptima de los elementos de un conjunto U a los elementos de un conjunto V. Así, el método propuesto por Justice and Hero (2006) determina la disimilaridad de dos grafos encontrando las correspondencias (o asignaciones óptimas) de los vértices entre estos grafos. Aunque este método posee una complejidad polinomial, su principal desventaja radica en no considerar la información de las aristas para el cálculo de la disimilaridad.

Recientemente, en la literatura, se han propuesto diversas soluciones subóptimas a la distancia de edición con una complejidad polinomial. En (Riesen and Bunke, 2009a) por ejemplo, se propone el cálculo de la GED basado en el problema de asignación. La idea de esta solución consiste en construir una matriz de costos C, como la mostrada en la figura 3.2, para encontrar las correspondencias de los vértices entre los grafos a comparar. Adicionalmente, en esta matriz C se considera la información de las aristas para obtener una mejor aproximación de la disimilaridad entre grafos.

Dados los grafos $g_1 = (V_1, E_1, \mu_1, \nu_1)$ y $g_2 = (V_2, E_2, \mu_2, \nu_2)$, la matriz de costos

C de tamaño $(n+m)\times(m+n)$ (donde n y m son el número de vértices de g_1 y g_2 respectivamente) está conformada de cuatro secciones, las cuales contienen las operaciones de edición -sustitución, inserción y eliminación- de los vértices entre ambos grafos (ver sección 2.3.1 para notación). La sección superior izquierda de C representa el costo de todas las posibles sustituciones de los vértices $(u_i \to v_j)$ entre g_1 y g_2 (donde $u_i \in V_1$ y $v_j \in V_2$), la diagonal de la parte superior derecha representa el costo de todas las posibles eliminaciones de los vértices $(u_i \to \epsilon)$ en g_1, y la diagonal de la parte inferior izquierda representa el costo de todas las posibles inserciones de vértices $(\epsilon \to v_j)$ en g_2. En esta matriz cada vértice puede ser eliminado o insertado una sola vez, por lo que cualquier elemento que no se encuentre en la diagonal de la parte superior derecha o inferior izquierda es asignada a ∞ (infinito). La parte inferior derecha de la matriz de costos es asignada a ceros pues sustituciones de la forma $(\epsilon \to \epsilon)$ no causan ningún costo.

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \cdots & c_{1m} & c_{1\varepsilon} & \infty & \cdots & \infty \\ c_{21} & c_{22} & \cdots & c_{2m} & \infty & c_{2\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n1} & c_{n2} & \cdots & c_{nm} & \infty & \cdots & \infty & c_{n\varepsilon} \\ \hline c_{\varepsilon 1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon m} & 0 & \cdots & 0 & 0 \end{array}\right]$$

Figura 3.2: Matriz de costos $C_{(n+m)\times(m+n)}$, donde cada entrada c_{ij} denota el costo por sustitución $c(u_i \to v_j)$, $c_{i\epsilon}$ el costo por eliminar un vértice $c(u_i \to \epsilon)$ y $c_{\epsilon j}$ el costo por insertar un vértice $c(\epsilon \to v_j)$ $(u_i \in V_1$ y $v_j \in V_2)$.

Una vez definida la matriz de costos de la figura 3.2, el algoritmo Munkres para el problema de asignación (Munkres, 1957) es aplicado para determinar la asignación de costo mínimo de los vértices de g_1 (representados por los renglones de C) a los vértices de g_2 (representados por las columnas de C); en otras palabras, la disimilaridad entre los grafos g_1 y g_2 es calculada. Con la finalidad de lograr una mejor aproximación de la disimilaridad, el método propuesto por Riesen and Bunke (2009a) añade a C los costos de las operaciones de edición de las aristas. En cada

entrada c_{ij} de la parte superior izquierda de C, se añade la suma mínima de los costos de las operaciones de edición de las aristas adyacentes a los vértices u_i y v_j. Esta suma mínima es calculada luego de construir una matriz de costos de manera similar a la de la figura 3.2. Los resultados experimentales muestran que el método propuesto por Riesen and Bunke (2009a) obtiene una buena aproximación de la disimilaridad entre grafos con respecto a la solución óptima propuesta por Berretti et al. (2001).

Una serie de mejoras han sido propuestas para el enfoque presentado por Riesen and Bunke (2009a). En (Fankhauser et al., 2011), por ejemplo, se propone el uso del algoritmo Volgenant-Jonker en lugar del algoritmo de Munkres para resolver el problema de asignación. El cálculo de la distancia de edición mediante el algoritmo Munkres muestra una complejidad de $O(n^4)$ de acuerdo a Fankhauser et al. (2011), mientras que el algoritmo Volgenant-Jonker tiene una complejidad de $O(n^3)$ (donde n es el tamaño de la matriz de costos).

Otra extensión al enfoque propuesto por Riesen and Bunke (2009a) puede ser encontrada en (Serratosa, 2014). La idea principal de esta extensión consiste en definir una matriz de costos de menor tamaño para solucionar el problema de asignación. La redefinición de esta matriz reduce el tiempo de ejecución del algoritmo utilizado para resolver el problema de asignación.

Recientemente, varias extensiones en (Riesen and Bunke, 2015) han sido desarrolladas con la finalidad de obtener una mejor aproximación a la distancia de edición entre grafos. Los resultados experimentales muestran que estas estrategias obtienen mejores resultados en determinados dominios; no obstante, el tiempo de ejecución se ve incrementado y en algunos casos los métodos propuestos por Riesen and Bunke (2015) llegan a ser inaplicables.

En la literatura, también se ha propuesto la distancia de edición de Hausdorff (Fischer et al., 2015). Este enfoque determina una aproximación de la distancia de edición combinando la idea del método propuesto por Riesen and Bunke (2009a) y una correspondencia entre vértices más eficiente mediante el uso de la distancia de Hausdorff. Aunque este enfoque tiene una complejidad cuadrática, este método es funcional sólo para algunos dominios de aplicación.

De los métodos previamente descritos, la correspondencia exacta se caracteriza por tener una complejidad exponencial. Por otra parte, en la correspondencia inexacta es posible encontrar trabajos basados en algoritmos genéticos que tienen el problema de ser no deterministas, o bien basados en *kernels de grafos* que son muy costosos. Además, la gran mayoría de estos trabajos están restringidos a un tipo de grafos en específico. Por otra parte, en la literatura existen trabajos como la Distancia de Edición de Grafos que no están restringidos a un tipo de grafos en específico y que permiten calcular la disimilaridad entre grafos con una complejidad polinomial.

3.4. Discusión

En la literatura, no se han reportado algoritmos diseñados para la selección de instancias en bases de datos de grafos. Los métodos existentes para la selección de instancias han sido desarrollados sólo para representaciones n-dimensionales.

Por otro lado, aunque aparentemente los trabajos de *selección de prototipos para embedding de grafos* seleccionan instancias, estos algoritmos no están diseñados para solucionar el problema planteado en esta tesis. El propósito de estos trabajos consiste en seleccionar elementos útiles del conjunto de entrenamiento para generar el espacio de representación, basado en disimilaridades, con una dimensionalidad reducida. En este nuevo espacio, todos los elementos del conjunto de entrenamiento son utilizados para clasificar.

Cabe señalar que los autores de la representación por disimilaridades (Pekalska et al., 2006; Riesen et al., 2007) muestran que incluso generando la representación con una selección aleatoria de prototipos, se obtienen buenos resultados de clasificación. No obstante, de acuerdo a los resultados experimentales reportados por los autores, una selección adecuada de los prototipos en el conjunto de entrenamiento permite mejorar la calidad de clasificación, siendo las mejores estrategias aquellas basadas en selección uniforme. Por otro lado, en la sección 5.2 de resultados experimentales se incluye este tipo de algoritmos, en la cual nos comparamos contra el mejor algoritmo

para la *selección de prototipos para embedding de grafos* reportado en la literatura, observando que este tipo de algoritmos no tienen un buen desempeño como selectores de instancias.

Por todo lo anterior, el desarrollo de un algoritmo de selección de instancias para bases de datos de grafos, que permita reducir el conjunto de entrenamiento, manteniendo, tanto como sea posible, la calidad de clasificación del conjunto de entrenamiento original es una línea de investigación abierta.

Capítulo 4

Algoritmos propuestos

En este capítulo se proponen diferentes soluciones al problema de la selección de instancias en bases de datos de grafos bajo tres enfoques. Primero, en la sección 4.1 se introducen tres algoritmos de selección propuestos en el espacio de los grafos. Segundo, en la sección 4.2 se explican las soluciones propuestas en el espacio n-dimensional generado por el *embedding de grafos vía disimilaridad*. Tercero, en la sección 4.3 se proponen algoritmos de selección híbridos desarrollados a partir de una combinación de los dos primeros enfoques. Cada algoritmo propuesto en este capítulo busca reducir el conjunto de entrenamiento, manteniendo, tanto como sea posible, la calidad de clasificación que puede obtenerse con el conjunto de entrenamiento original.

4.1. Selección de instancias en el espacio de los grafos

El primer enfoque, como se muestra en la figura 4.1, consiste en realizar una selección de instancias directamente en el espacio de los grafos, usando una medida de disimilaridad entre grafos. Así, el conjunto de entrenamiento es reducido y usado para clasificar nuevos grafos en el espacio de representación de los mismos.

En las siguientes secciones se detallan los tres algoritmos de selección de instancias propuestos para este enfoque. Los dos primeros algoritmos están basados en estrategias de selección de tipo *wrapper* y el tercero en una estrategia de tipo $filter$.

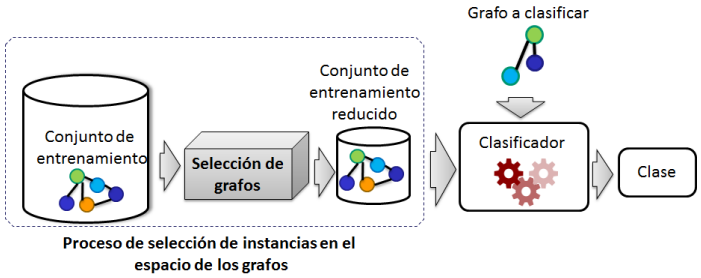

Figura 4.1: Proceso de selección de instancias en el espacio de los grafos.

4.1.1. DROPRel

El primer algoritmo de selección de instancias propuesto, de tipo *wrapper*, es una extensión de los algoritmos DROP 1-5 (*Decremental Reduction Optimization Procedure*) diseñados para espacios n-dimensionales (Wilson and Martinez, 2000). La idea general de estos algoritmos consiste en descartar aquellas instancias del conjunto de entrenamiento que no afectan la calidad de clasificación. La primera versión, denominada DROP1, no toma en cuenta el orden en el que se procesan dichas instancias, a diferencia de DROP2 que procesa primero aquellas que están ubicadas en el interior de la clase, terminando con las de la frontera de decisión. En el caso de los algoritmos DROP3-5, éstos usan el mismo orden de procesamiento que DROP2, pero aplican previamente un filtro de ruido. DROP3-5 se diferencian en el filtro utilizado.

Nuestra extensión, denominada DROPRel, inicia asignando el conjunto de entrenamiento T al subconjunto a seleccionar S (ver pseudocódigo en el algoritmo *1*). Luego, DROPRel construye una lista de los grafos vecinos menos disimilares y un listado de socios para cada grafo de S (líneas 3-5). La disimilaridad entre dos grafos es cuantificada usando la distancia de edición propuesta por Fankhauser et al. (2011); Riesen and Bunke (2009a), mientras que los socios de un grafo g son aquellos grafos que tienen a g como vecino menos disimilar. Posteriormente, los grafos en S son ordenados descendentemente con respecto a su disimilaridad promedio (líneas 6-7), la cual es calculada como sigue:

$$avgdisim(g) = \frac{\sum_{g' \in C, g \neq g'} d(g, g')}{|C| - 1} \tag{4.1.1}$$

donde:

C es el conjunto de grafos de la misma clase que g.

$d(g, g')$ es la distancia de edición de grafos.

Esta expresión calcula la disimilaridad promedio que tiene g con respecto a los demás grafos de su misma clase.

DROPRel procesa cada grafo g en S de acuerdo a su disimilaridad promedio (expresión 4.1.1), iniciando con los más disimilares (valores altos) (líneas 8-19). Si el grafo g no afecta la calidad de clasificación de sus socios, entonces g es descartado de S (líneas 11-18). Consecuentemente, todos los socios del grafo g eliminan a g de su lista de grafos vecinos menos disimilares. Luego, se encuentran los nuevos grafos vecinos menos disimilares para los socios que eliminaron a g de su lista, y estos socios son añadidos al listado de socios de cada nuevo vecino (líneas 14-16). La idea de mantener un listado de socios es para evitar clasificar todos los grafos en S, pues al eliminar g de S los únicos grafos afectados son aquellos que tienen a g como grafo vecino menos disimilar.

A diferencia de los algoritmos DROP, el algoritmo DROPRel, diseñado para el espacio de los grafos, descarta los elementos del conjunto de entrenamiento que no afectan la calidad de clasificación, analizando primero aquellos grafos más disimilares a los grafos de su clase y finalizando con los grafos menos disimilares a los grafos de su clase. Esta estrategia fue adoptada con el propósito de eliminar primero aquellos grafos mal etiquetados, dado que estos grafos son usualmente los más disimilares con los grafos de su clase; así evitamos aplicar un filtro de ruido previo a este análisis.

Hay que notar que el primer criterio de selección es la calidad de clasificación. Así, incluso aquellos grafos menos disimilares que a los grafos de su clase pueden ser descartados del conjunto S si estos no afectan el desempeño de la clasificación.

Algoritmo 1: $DROPRel(T)$

Entrada: $T = (g_1, \ldots, g_m)$: Conjunto de entrenamiento;
Salida: $S = (s_1, \ldots, s_n)$: Conjunto de grafos seleccionadas;

```
1  S ← T;
2  Para cada grafo g en S hacer
3  |  g.nn ← los k + 1 grafos vecinos menos disimilares de g en S
4  |  Añadir g a cada una las listas de socios de sus grafos vecinos menos
   |  disimilares
5  fin
6  CalcularDisimilaridadPromedio(S)
7  OrdenarPorDisimilaridadPromedio(S)
8  Para cada grafo g en S ordenado por su disimilaridad promedio hacer
9  |  conp ← número de socios de g clasificados correctamente con g como
   |  vecino menos disimilar
10 |  sinp ← número de socios de g clasificados correctamente sin g como vecino
   |  menos disimilar
11 |  Si sinp ≥ conp entonces
12 |  |  Descartar g de S
13 |  |  Para cada socio A de g hacer
14 |  |  |  Descartar g de la lista de grafos vecinos menos disimilares de A
15 |  |  |  Encontrar un nuevo grafo vecino menos disimilar para A
16 |  |  |  Añadir A a la lista de socios del nuevo grafo vecino
17 |  |  fin
18 |  fin
19 fin
```

Sin embargo, el orden en el que se procesan los grafos, en nuestro caso el inducido por el promedio de disimilaridad, es importante para obtener un mayor grado de generalización y por ende una mejor calidad de clasificación.

4.1.2. DROPRel+FSS

El segundo algoritmo propuesto para la selección de instancias en el espacio de los grafos consiste en un ensamble de selectores. Este ensamble está formado por dos etapas: una de *exclusión* y una de *inclusión*.

La etapa de *exclusión* tiene como propósito obtener un conjunto reducido S basado en una estrategia de selección decremental. En esta etapa es utilizado el

algoritmo de selección de instancias DROPRel, propuesto en la sección anterior.

Por otra parte, la etapa de *inclusión* tiene la finalidad de analizar de manera secuencial si la incorporación de alguno de los grafos descartados por DROPRel, en el conjunto seleccionado, contribuye a mejorar la calidad de clasificación. Este análisis secuencial es conocido como *Forward Sequential Selection* (FSS) (Olvera-López et al., 2009). El pseudocógido del algoritmo propuesto en esta sección es mostrado en el *Algoritmo 2*.

Algoritmo 2: $DROPRel + FSS(T)$

Entrada: $T = (g_1, \ldots, g_m)$: Conjunto de entrenamiento;
Salida: $S = (s_1, \ldots, s_n)$: Conjunto de grafos seleccionados;

```
1  S ← DROPRel(T);
2  D ← T − S //grafos descartados
3  mejorClasif ← Clasificar(S)
4  Para cada grafo g en D hacer
5  |  S' ← S ∪ {g};
6  |  Si Clasificar(S') > mejorClasif entonces
7  |  |  mejorClasif ← Clasificar(S')
8  |  |  S ← S ∪ {g}
9  |  fin
10 fin
```

4.1.3. ExtPSC

La tercera solución propuesta es una extensión al algoritmo PSC (*Prototype Selection by Clustering*) presentado por Olvera-López et al. (2010) para espacios n-dimensionales. La idea de este algoritmo, de tipo $filter$, consiste en retener instancias ubicadas en la frontera de decisión y algunas al interior de la clase. Para este propósito, PSC genera un cierto número de grupos y determina si son homogéneos o no. Un grupo es homogéneo si contiene instancias de una sola clase, mientras que un grupo no homogéneo es aquel que contiene instancias de dos o más clases. Si el grupo es homogéneo PSC sólo retiene el centro del grupo, en caso contrario PSC selecciona las instancias frontera (instancias más cercanas entre sí de clase diferente) de los grupos no homogéneos.

Nuestra extensión, denominada ExtPSC (*Extension of Prototype Selection by Clustering*), ha sido diseñada para el espacio de los grafos usando la distancia de edición (Fankhauser et al., 2011; Riesen and Bunke, 2009a) como medida de disimilaridad entre grafos. Un cierto número de grupos es creado utilizando un algoritmo similar a K-means, en el cual el centro de cada grupo es el grafo menos disimilar a los demás grafos del grupo. Posteriormente, los grupos son divididos en homogéneos (conteniendo grafos de una sola clase) y no homogéneos (conteniendo grafos de más de una clase). Si el grupo es homogéneo seleccionamos el centro de cada grupo, en caso contrario cuando el grupo es no homogéneo se determinan los grafos ubicados en la frontera de decisión de las clases. Para determinar el grafo frontera de un grafo g en un grupo no homogéneo, seleccionamos aquel grafo g_f menos disimilar de clase diferente a g, procurando que g_f se encuentre lo más disimilar posible de regiones que pertenecen a grafos de otras clases. Esta mejora surge a partir de que PSC necesita una frontera de decisión bien definida para llevar a cabo una buena selección, pues, de otra manera, este algoritmo selecciona aquellos grafos menos disimilares entre sí de clase diferente; provocando que el conjunto seleccionado final, no obtenga tampoco una frontera de decisión bien definida (ver figura 4.2(b)). En la figura 4.2 es posible observar un ejemplo de la selección de instancias realizada por PSC y ExtPSC.

El algoritmo ExtPSC inicia dividiendo el conjunto de entrenamiento T en C grupos. Originalmente en (Olvera-López et al., 2010), K-Means es usado para realizar esta actividad; sin embargo, en el espacio de los grafos no es posible aplicar directamente este algoritmo, pues el principal problema radica en cómo calcular los centros de cada grupo en cada iteración. Para nuestro caso en particular, el centro de un grupo A_j se determina como sigue:

$$centro(A_j) = arg \min_{g_1 \in A_j} \sum_{g_2 \in A_j} d(g_1, g_2) \tag{4.1.2}$$

donde:

$d(g_1, g_2)$ es la distancia de edición de grafos.

La idea de esta expresión consiste en localizar un grafo $g_1 \in A_j$ tal que la suma de las distancias de edición de g_1 con respecto a todos los otros grafos en A_j sea mínima. En la literatura, esta expresión es conocida como el *grafo mediana* (Jiang et al., 2001).

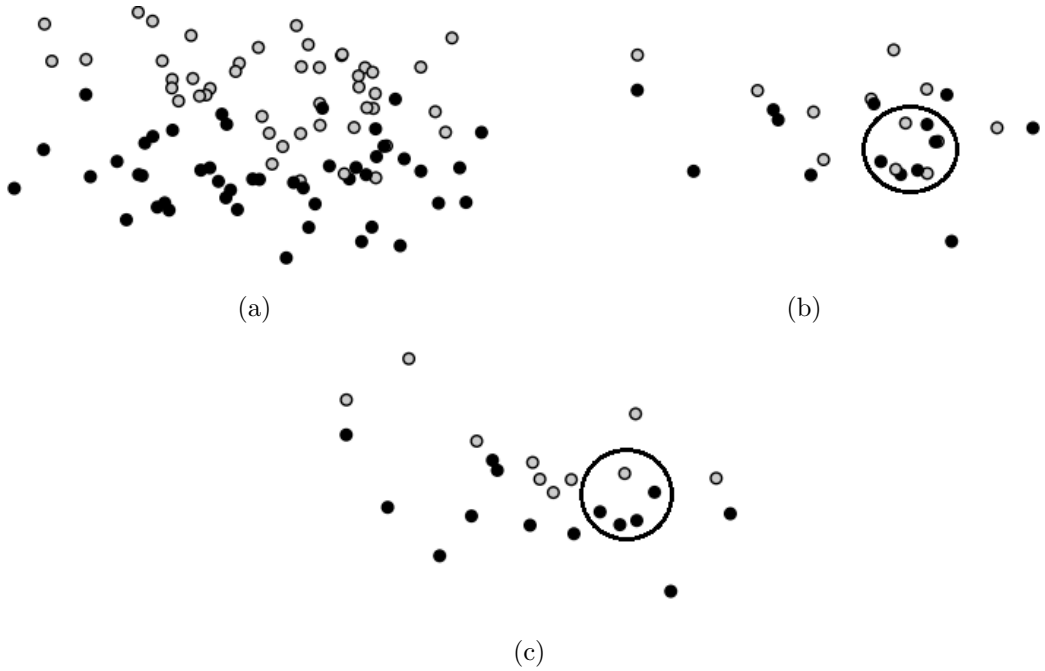

Figura 4.2: a) Conjunto artificial para dos clases; (b) Conjunto seleccionado por PSC; (c) Conjunto seleccionado por ExtPSC.

ExtPSC analiza, de manera similar a PSC, los C grupos generados, con el fin de determinar para cada grupo si es homogéneo o no. El pseudocódigo de ExtPSC es mostrado en el *Algoritmo* 3.

Un grupo A_j es homogéneo si contiene grafos que pertenecen a una sola clase. En este caso, ExtPSC retiene el centro del grupo y los grafos restantes dentro de este grupo son descartados (líneas 4-7).

Por otro lado, si un grupo A_j es no homogéneo, éste contiene grafos de dos o más clases. Así, ExtPSC analiza los grafos de A_j para determinar aquellos situados cerca de la frontera de decisión de cada clase (líneas 9-18). En esta etapa del algoritmo, ExtPSC aplica primero un filtro ENN (Wilson, 1972) para identificar los grafos ruidosos (línea 9), evaluando cada grafo del grupo A_j sobre todo el conjunto de entrenamiento original T. Hay que notar que este filtro de ruido puede ser aplicado al inicio de ExtPSC, tal y como es utilizado normalmente por varios algoritmos de selección de instancias, evaluando cada elemento del conjunto T sobre el mismo

Algoritmo 3: $ExtPSC(T, C)$

Entrada: $T = (g_1, \ldots, g_m)$: Conjunto de entrenamiento; C: Número de grupos
Salida: $S = (s_1, \ldots, s_n)$: Conjunto de instancias seleccionadas

```
1  S ← ∅;
2  Clusters ← K-MedianGraph(T, C);
3  Para cada grupo A_j en Clusters hacer
4      Si A_j es homogéneo entonces
5          c ← el centro de A_j;
6          S ← S ∪ {c};
7      fin
8      sino
9          G_R ← ENN(A_j, T); //conjunto de grafos ruidosos en A_j
10         C_M ← la clase mayoritaria de A_j;
11         Para cada clase C_K en A_j, donde C_K ≠ C_M hacer
12             Para cada grafo g_i que pertenece a la clase C_K hacer
13                 Sea g_c ∈ C_M, g_c ∉ G_R el grafo menos disimilar a g_i, procurando
                   que g_c sea lo más disimilar posible a los grafos de otras clases;
14                 S ← S ∪ {p_c};
15                 Sea g_M ∈ C_K, g_M ∉ G_R el grafo menos disimilar a g_c, procurando
                   que g_M sea lo más disimilar posible a los grafos de otras clases;
16                 S ← S ∪ {g_M};
17             fin
18         fin
19     fin
20 fin
```

conjunto. Sin embargo, la idea de postergar la aplicación de este filtro hasta esta etapa es para evitar trabajo innecesario, ya que en esta etapa no es indispensable evaluar aquellos grafos que pertenecen a grupos homogéneos, pues sólo el centroide es retenido y el resto es descartado. Finalmente, es importante señalar que los grafos ruidosos G_R sólo son identificados pero no eliminados del conjunto de entrenamiento T ni del grupo A_j, pues aunque no se consideran para la selección de un grafo frontera g_c, sí se utilizan para calcular la disimilaridad promedio que existe entre g_c y los grafos de otras clases en A_j; este procedimiento es descrito en el resto de esta sección.

ExtPSC encuentra para un grupo no homogéneo A_j la clase más frecuente C_M

(clase mayoritaria) (línea 10). Posteriormente, el algoritmo ExtPSC determina los grafos situados cerca de la frontera de decisión de cada clase (líneas 12-17). Así, para cada grafo $g_i \in C_k$ se determina su grafo frontera g_c en la clase mayoritaria C_M (líneas 12-13), usando la siguiente función:

$$f_g(g_c) = \underset{g_c \in C_M, g_c \notin G_R}{\operatorname{argmin}} \frac{d(g_c, g_i)}{avgdisim(g_c)} \tag{4.1.3}$$

donde:

C_M es la clase mayoritaria.
$d(g_c, g_i)$ es la distancia de edición de grafos.
$avgdisim(g_c)$ es la disimilaridad promedio entre g_c y los grafos de otras clases.

La función $avgdisim(g_c)$ está dada por:

$$avgdisim(g_c) = \frac{\sum_{g_j \in \overline{C_M}} d(g_j, g_c)}{|\overline{C_M}|} \tag{4.1.4}$$

donde:

$\overline{C_M}$ es el conjunto de grafos de clase diferente a C_M.
$d(g_j, g_c)$ es la distancia de edición de grafos.

La expresión 4.1.3 encuentra un grafo frontera $g_c \in C_M, g_c \notin G_R$ tal que la razón entre la disimilaridad $d(g_c, g_i)$ y el promedio de disimilaridad de g_c con respecto a los grafos de otras clases sea mínima. En otras palabras, la idea es encontrar un grafo frontera g_c lo menos disimilar a g_i, intentando que g_c sea lo más disimilar posible a los grafos de otras clases en A_j. Si $avgdisim(g_c)$ obtiene valores bajos, esto indica que g_c no es muy disimilar a los grafos de clase diferente, mientras que valores altos indican que es muy disimilar. En lo referente al grafo frontera g_M de la clase C_K correspondiente al grafo g_c (línea 15), éste es determinado de la misma manera.

El algoritmo ExtPSC selecciona los grafos más representativos (centros) para cada grupo homogéneo y los grafos ubicados cerca de la frontera de decisión de los grupos no homogéneos.

4.2. Selección de instancias en el espacio n-dimensional

El segundo enfoque de selección, como se presenta en la figura 4.3, consiste en obtener primero la representación del conjunto de entrenamiento en el espacio n-dimensional mediante el *embedding de grafos vía disimilaridad* (sección 3.2). Posteriormente, un algoritmo de selección de instancias diseñado para espacios n-dimensionales puede ser aplicado sobre la nueva representación del conjunto de entrenamiento, obteniendo así un conjunto reducido. Este conjunto de entrenamiento reducido es utilizado para clasificar nuevos grafos en el espacio n-dimensional. Para este propósito, un nuevo grafo a clasificar tiene que ser transformado a un vector de características de n-dimensiones.

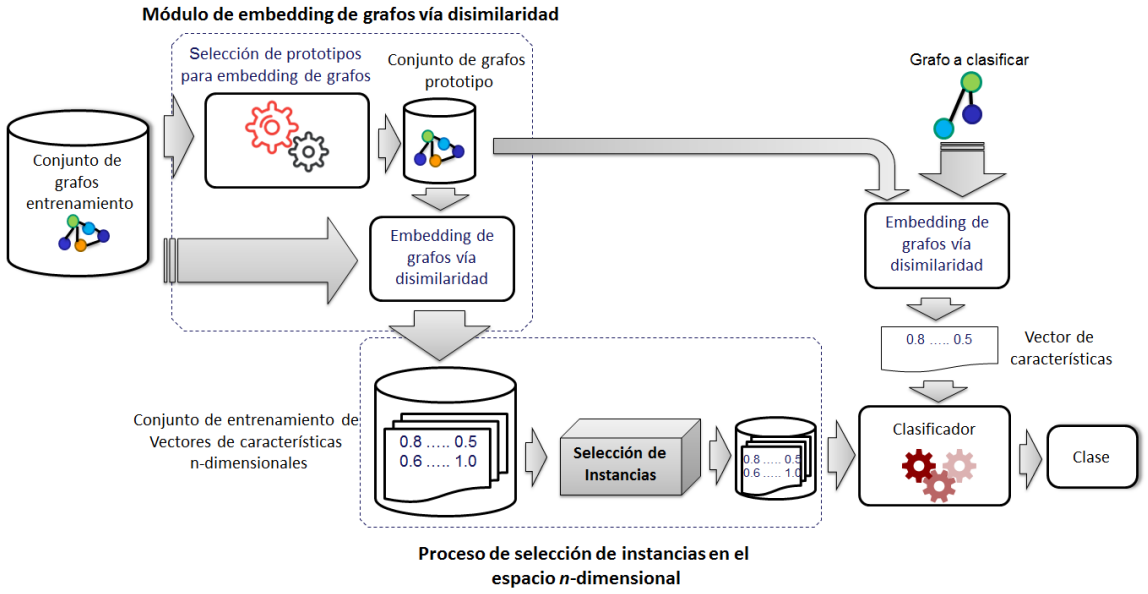

Figura 4.3: Proceso de selección de instancias en el espacio n-dimensional generado por el embedding de grafos vía disimilaridad.

En la siguiente sección, se describen los algoritmos de selección de instancias propuestos para el espacio de representación n-dimensional generado por el *embedding de grafos vía disimilaridad.*

4.2.1. DsDROPRel, DsDROPRel+FSS y DsExtPSC

En esta sección se proponen los algoritmos de selección de instancias DsDROPRel, DsDROPRel+FSS y DsExtPSC, mismos que son adaptaciones, en el espacio n-dimensional, de los algoritmos propuestos en las secciones 4.1.1, 4.1.2 y 4.1.3. La idea de estas nuevas propuestas consiste en utilizar la misma estrategia de selección de instancias diseñada para el espacio de los grafos, pero ahora en el espacio n-dimensional. Por lo que la principal diferencia, con respecto a los algoritmos originalmente propuestos en el espacio de los grafos, radica en la medida de disimilaridad utilizada. Para este propósito, se utilizó la medida HVDM propuesta por Wilson and Martinez (2000), dado que los algoritmos de referencia en nuestra comparación experimental (sección 5.3) hacen uso de esta medida.

4.3. Selección de instancias híbrida

La idea de la selección híbrida, mostrada en la figura 4.4, consiste primero en realizar una selección de instancias en el espacio de los grafos, usando un algoritmo diseñado para este fin (sección 4.1). Posteriormente, el conjunto seleccionado de instancias es usado como conjunto de prototipos en el *embedding de grafos vía disimilaridad* (sección 3.2) para generar el espacio de representación n-dimensional del mismo conjunto y del conjunto de instancias descartadas por la selección en el espacio de los grafos. Finalmente, se analiza en el espacio n-dimensional si la inclusión de algunas de las instancias descartadas contribuye a mejorar la calidad de clasificación. El conjunto final seleccionado está conformado por las instancias seleccionadas en el espacio de los grafos y las instancias adicionadas en el espacio n-dimensional.

De manera similar al enfoque de selección de instancias de la sección 4.2, el conjunto final seleccionado es utilizado para clasificar nuevos grafos en el espacio n-dimensional. Por lo tanto, un nuevo grafo a clasificar tiene que ser transformado a este espacio de representación para su clasificación.

En las siguientes secciones se describen dos algoritmos de selección de instancias híbridos con base en el proceso de selección descrito anteriormente.

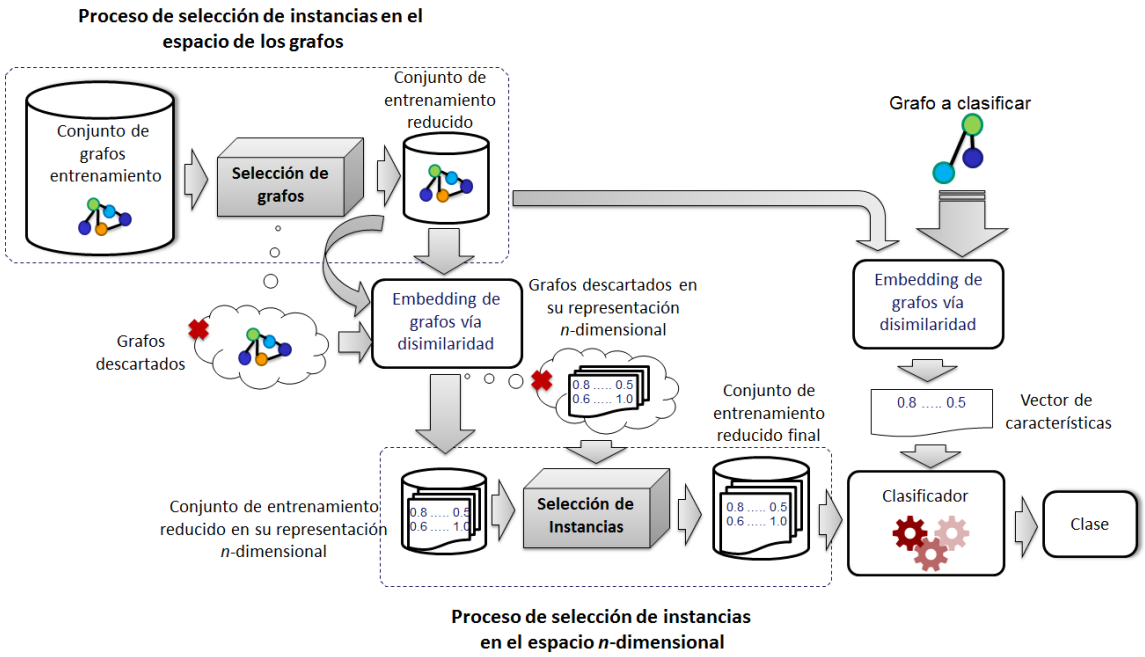

Figura 4.4: Proceso de selección de instancias híbrida.

4.3.1. HDroprel+FSS

El primer algoritmo de selección de instancias híbrido es denominado HDroprel+FSS y un pseudocódigo del mismo es mostrado en el *Algoritmo 4*. La idea de esta propuesta consiste en aplicar primero DROPRel (sección 4.1.1) en el espacio de los grafos (línea 1). Luego, el conjunto seleccionado S por DROPRel es utilizado como conjunto de prototipos en el *embedding de grafos vía disimilaridad* para generar la representación n-dimensional del mismo conjunto y del conjunto D de grafos descartados por DROPRel (líneas 3-6). En esta primera etapa de selección, se tiene como propósito obtener un conjunto de entrenamiento reducido útil para la clasificación, así como un conjunto que pueda ser utilizado en el *embedding de grafos vía disimilaridad*.

Posteriormente, en el espacio n-dimensional, se analiza, de manera secuencial mediante el algoritmo *Forward Sequential Selection* (FSS) (Olvera-López et al., 2009), si la inclusión de alguna de las instancias descartadas por DROPRel ayuda a mejorar la calidad de clasificación (líneas 7-14).

Algoritmo 4: $HDroprel + FSS(T)$

Entrada: $T = (g_1, \ldots, g_m)$: Conjunto de entrenamiento de grafos;
Salida: $S_{sim} = (s_1, \ldots, s_n)$: Conjunto de instancias seleccionadas en el espacio n-dimensional;

```
S ← DROPRel(T);//grafos seleccionados
D ← T − S //grafos descartados
//Embedding de S usando el conjunto mismo
S_sim ← embeddingViaDisimilaridad(S, S);
//embedding de D usando el conjunto S
D_sim ← embeddingViaDisimilaridad(D, S);
mejorClasif ← Clasificar(S_sim)
Para cada instancia p en D_sim hacer
    S'_sim ← S_sim ∪ {p};
    Si Clasificar(S'_sim) > mejorClasif entonces
        mejorClasif ← Clasificar(S'_sim)
        S_sim ← S_sim ∪ {p}
    fin
fin
```

4.3.2. HExtPSC+FSS

En el segundo algoritmo de selección de instancias híbrido utilizamos la misma idea a la del algoritmo presentado en la sección anterior. La diferencia en esta nueva propuesta radica en el algoritmo de selección de instancias aplicado en el espacio de los grafos. Para esta nueva propuesta se utilizó el algoritmo ExtPSC (ver sección 4.1.3).

De la misma manera que el algoritmo de la sección 4.3.1, el conjunto S seleccionado por ExtPSC es usado como conjunto de prototipos para generar el espacio de representación n-dimensional, mediante el *embedding de grafos vía disimilaridad*. En este nuevo espacio de representación, se utiliza el algoritmo FSS para analizar si la incorporación de alguna de los instancias descartadas por ExtPSC mejoran la calidad de clasificación.

4.4. Discusión

La idea de explorar diferentes enfoques de selección es encontrar la mejor solución al problema planteado en esta tesis. Primero, en el espacio de los grafos se propusieron dos extensiones basadas en estrategias de selección de instancias originalmente diseñadas para espacios n-dimensionales, las cuales fueron denominadas DROPRel y ExtPSC. Además DROPRel se combina con un proceso para adicionar aquellos grafos que fueron descartados en una primera etapa y que permiten mejorar la calidad del conjunto de entrenamiento.

Por otra parte, las estrategias de selección propuestas en el espacio de los grafos fueron utilizadas sobre el espacio n-dimensional generado por el *embedding de grafos vía disimilaridad*. Cabe señalar que estas estrategias ofrecen una mejora incluso que soluciones como los algoritmos DROPs y PSC, lo cual se muestra en la sección 4.3.

Por último, se proponen estrategias híbridas que seleccionan instancias dentro del espacio de los grafos y posteriormente, en el espacio n-dimensional generado por el *embedding de grafos vía disimilaridad*, se seleccionan aquellas instancias descartadas en el espacio de los grafos que contribuyen a mejorar la calidad de clasificación.

Capítulo 5

Resultados experimentales

En este capítulo se presentan los resultados experimentales obtenidos al aplicar los algoritmos propuestos en esta tesis sobre distintas bases de datos de grafos. Primero, en la sección 5.1 se describe la configuración utilizada en los experimentos. Posteriormente, en las secciones 5.2, 5.3 y 5.4, se muestra para cada enfoque de selección de instancias los resultados obtenidos por los algoritmos propuestos y otros algoritmos de referencia. Finalmente, en la sección 5.5 se presenta una comparación entre los algoritmos de selección de instancias propuestos.

5.1. Descripción de los experimentos

Los algoritmos presentados en el capítulo anterior fueron evaluados sobre diez bases de datos de grafos obtenidas del repositorio *IAM Graph Database* de la Universidad de Berna, Suiza (Riesen and Bunke, 2008). Este repositorio fue seleccionado para nuestra evaluación experimental puesto que es uno de los repositorios de grafos más utilizados en las experimentaciones reportadas en la literatura. En la tabla 5.1 se describen las principales características de cada base de datos utilizada, especificando el número de grafos, número de clases, así como el número máximo y número promedio de vértices y aristas por grafo.

En cada evaluación experimental las bases de datos fueron divididas en conjuntos de entrenamiento y prueba, usando validación cruzada estratificada de diez pliegues (*stratified 10 fold cross validation*). Este tipo de validación es una de las técnicas

Tabla 5.1: Resumen de las características de las bases de datos de grafos utilizadas en nuestros experimentos. Se muestra el número de grafos, número de clases, número promedio y número máximo de vértices y aristas.

Base de datos	**No. Grafos**	**No. Clases**	**No. vértices promedio**	**No. aristas promedio**	**No. vértices máximo**	**No. aristas máximo**
Letter low	2250	15	4.7	3.1	8	6
Letter medium	2250	15	4.7	3.2	9	7
Letter high	2250	15	4.7	4.5	9	9
GREC	1100	22	11.5	12.2	25	30
Fingerprints	2800	4	5.4	4.4	26	24
Protein	600	6	32.6	62.1	126	149
AIDS	2000	2	15.7	16.2	95	103
Mutagenicity	3500	2	30.3	30.8	417	112
Coil-RAG	3900	100	3	3	11	13
Coil-DEL	3900	100	21.5	54.2	77	222

más comúnmente utilizadas para evaluar resultados de clasificación.

Una vez que se obtuvieron los respectivos conjuntos de entrenamiento y prueba, el primero de éstos es utilizado por los algoritmos de selección de instancias para reducir el conjunto mismo de acuerdo cada enfoque de selección (ver capítulo 4). Posteriormente, la calidad de clasificación del conjunto seleccionado es evaluada con el conjunto de prueba, usando el clasificador *k-Nearest Neighbor* (k-NN) junto a una medida de disimilaridad. Este clasificador es uno de los más utilizados en grafos dada su simplicidad de aplicación. Cabe señalar que para la clasificación en el espacio de representación de los grafos se utilizó la distancia de edición (Fankhauser et al., 2011; Riesen and Bunke, 2009a) como medida de disimilaridad, mientras en el espacio n-dimensional se optó por la distancia Euclidiana. No se omite mencionar que para la evaluación de cada algoritmo, se utilizaron los mismos conjuntos de entrenamiento y prueba, además del mismo equipo de cómputo, el cual consta de un procesador Intel Core i7 3.30 GHz con 32GB en RAM y un sistema operativo Linux Ubuntu 14.04.

Los experimentos reportados en esta tesis incluyen una evaluación en términos de la calidad de clasificación (*accuracy*), retención (porcentaje de grafos seleccionados) y tiempo de ejecución para cada algoritmo. Adicionalmente, para los resultados de clasificación y retención promedio reportados en cada tabla, se muestra una gráfica de comparación de los algoritmos de selección de instancias presentados. El mejor algoritmo en calidad de clasificación es el más alejado del eje horizontal de la gráfica de comparación, mientras que el mejor algoritmo en retención es el más cercano

a la vertical. En estas gráficas se muestran también los algoritmos que pertenecen al frente de Pareto, los cuales se indican con el símbolo "□". El frente de Pareto, utilizado para evaluar funciones multiobjetivo, está integrado por aquellos algoritmos que obtienen mejores o iguales resultados que todos los demás en al menos uno de los objetivos. Es decir, aquellos algoritmos que no son superados por ningún otro en todos los objetivos. En el contexto de la selección de instancias se consideran dos objetivos: accuracy y retención.

Finalmente, para validar los resultados de clasificación se utilizó la prueba de *Friedman* (Demšar, 2006), la cual determina si existe diferencia estadística significativa entre los algoritmos comparados. En los casos en los que se encontraron diferencias significativas se realizó el test estadístico *post-hoc Bergmann-Hommel*. Los resultados de este análisis estadístico son mostrados en diagramas de diferencia crítica (CD por sus siglas en inglés *Critical Difference*), los cuales presentan el ranking promedio de la calidad de clasificación obtenida, además de las diferencias significativas observadas. En estos diagramas, el algoritmo de selección de instancias más a la derecha es considerado el mejor y si dos algoritmos comparten un línea gruesa significa que tienen un comportamiento estadísticamente similar.

5.2. Resultados experimentales en el espacio de los grafos

En esta sección se presentan los resultados obtenidos para los algoritmos de tipo *wrapper* DROPRel, DROPRel+FSS y el algoritmo de tipo *filter* ExtPSC, propuestos en el espacio de los grafos. Primero, se menciona qué algoritmos, además de los propuestos, fueron utilizados en nuestra comparación. Posteriormente, se explican los experimentos realizados para ajustar los distintos parámetros de los algoritmos presentados. Finalmente, se muestran los resultados de la comparación experimental.

5.2.1. Algoritmos usados para la comparación

Como se mencionó en el capítulo 3, en la literatura no se reportan algoritmos diseñados para el problema de la selección de instancias en bases de datos de grafos. Los algoritmos más cercanos a este problema, como los de *selección de prototipos*

para embedding de grafos (ver sección 3.2), tienen el propósito de generar el espacio de representación n-dimensional, pero no reducen el conjunto de entrenamiento. Aunque todos los algoritmos de la sección 3.2 son directamente comparables, se seleccionó aquel trabajo que reporta los mejores resultados de clasificación. Este trabajo, propuesto por (Borzeshi et al., 2013), presenta un total de seis algoritmos para la *selección de prototipos para embedding de grafos*, de los cuales el algoritmo *Discriminative Spanning Prototype Selection* (D-SPS) produce los mejores resultados de clasificación de acuerdo a lo reportado por los autores, siendo este algoritmo el elegido para nuestra comparación.

Adicionalmente, se presenta una comparación con los algoritmos de tipo *wrapper* GsDROP3 y GsDROP5, así como el algoritmo de tipo *filter* GsPSC, los cuales son adaptaciones al espacio de los grafos, realizadas por nosotros, de los algoritmos de selección de instancias DROP3, DROP5 (Wilson and Martinez, 2000) y PSC (Olvera-López et al., 2010), originalmente diseñados para espacios n-dimensionales. Para las adaptaciones de estos algoritmos se usó la distancia de edición como medida de disimilaridad entre grafos (Fankhauser et al., 2011; Riesen and Bunke, 2009a). Además, en GsPSC, los centros de cada grupo son calculados mediante la expresión del *grafo mediana*, mostrada en la sección 4.1.2.

5.2.2. Ajuste de parámetros

Los algoritmos *wrapper* como DROPRel, DROPRel+FSS, GsDROP3 y GsDROP5, no usan algún parámetro para su ejecución. Por otro lado, los algoritmos *filter* ExtPSC y GsPSC, utilizan el número de grupos C a generar como único parámetro. Mientras que en el caso del algoritmo D-SPS, considerado de tipo *filter* por su estrategia de selección, el número de prototipos n por clase a seleccionar es necesario.

Con el objetivo de determinar el número de grupos C para los algoritmos GsPSC y ExtPSC, se hicieron experimentos con diferentes valores para este parámetro usando el clasificador 1-NN. Este clasificador fue elegido para estos experimentos dado que obtiene los mejores resultados de clasificación con el conjunto de entrenamiento

original (ver tabla 5.6). En las tablas 5.2, 5.3 y 5.4, por otro lado, se muestran los resultados de clasificación, retención y tiempo de ejecución obtenidos por GsPSC y ExtPSC, utilizando los valores $C = 6t, 8t, 10t$ y $12t$, donde t es el número de clases. Cabe señalar que los tiempos de ejecución fueron medidos posterior al tiempo que toma el cálculo de la matriz de distancias entre los grafos del conjunto de entrenamiento. Esta matriz es necesaria para la ejecución de todos los algoritmos de selección de instancias en el espacio de los grafos. Para los experimentos reportados en la siguiente sección se usó $C = 10t$, pues con base en los resultados mostrados de la tabla 5.2, los algoritmos GsPSC y ExtPSC obtienen en promedio los mejores resultados de clasificación usando este valor de C; manteniendo un buen balance con la retención y el tiempo de ejecución.

Tabla 5.2: Resultados de clasificación con 1-NN para los algoritmos GsPSC y ExtPSC, usando $C = 6t, 8t, 10t$ y $12t$.

		$C = 6t$		$C = 8t$		$C = 10t$		$C = 12t$	
Base de datos	**Orig.**	**GS PSC**	**ExtPSC**	**GS PSC**	**ExtPSC**	**GS PSC**	**ExtPSC**	**GS PSC**	**ExtPSC**
Letter-low	99.56	99.51	99.29	99.42	99.29	99.47	99.42	99.33	99.29
Letter-med	94.71	87.64	89.96	87.42	90.58	88.00	90.18	89.42	90.71
Letter-high	90.44	81.51	84.04	82.36	83.51	83.51	85.42	83.73	85.60
GREC	99.36	99.27	99.36	99.55	99.36	99.45	99.73	99.45	99.27
Fingerprint	80.54	73.18	79.89	73.71	80.21	73.89	81.11	72.50	80.11
AIDS	99.20	96.75	99.40	93.65	98.95	95.30	99.40	95.65	99.20
Protein	72.33	41.67	57.83	47.00	56.00	49.67	57.17	47.67	57.17
Mutagenicity	70.81	61.29	67.84	60.85	69.01	61.63	68.50	61.24	66.57
Coil-rag	96.90	92.82	92.46	93.38	93.33	94.10	93.38	94.90	94.08
Coil-del	94.69	82.05	85.97	84.72	86.54	87.79	88.05	88.77	88.41
Promedio	89.85	81.57	85.61	82.21	85.68	**83.28**	**86.24**	83.27	86.04

Tabla 5.3: Resultados de retención obtenidos con GsPSC y ExtPSC, usando $C = 6t, 8t, 10t$ y $12t$.

		$C = 6t$		$C = 8t$		$C = 10t$		$C = 12t$	
Base de datos	**Orig.**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**
Letter-low	100.00	4.80	4.98	6.55	6.64	7.98	7.81	9.53	9.42
Letter-med	100.00	13.46	16.89	14.39	16.03	14.94	18.00	15.27	18.19
Letter-high	100.00	19.43	29.00	19.74	27.08	19.60	25.00	21.72	26.93
GREC	100.00	14.25	14.61	18.25	18.27	22.86	22.62	27.35	27.45
Fingerprint	100.00	23.62	21.31	23.21	21.10	23.12	21.00	22.88	21.63
AIDS	100.00	6.99	8.53	3.49	4.76	4.44	9.00	3.66	5.66
Protein	100.00	15.80	35.69	19.22	34.22	21.39	38.48	22.48	39.19
Mutagenicity	100.00	33.67	32.96	33.88	33.48	33.56	33.44	33.28	33.43
Coil-rag	100.00	25.66	29.56	30.12	31.56	34.21	35.27	38.97	39.28
Coil-del	100.00	33.13	42.81	36.32	41.94	39.46	43.66	43.04	44.78
Promedio	100.00	**19.08**	23.63	20.52	**23.51**	22.16	25.43	23.82	26.60

Tabla 5.4: Tiempo por calcular la matriz de distancias entre grafos para cada base de datos y tiempos de ejecución promedio obtenidos con GsPSC y ExtPSC, usando $C = 6t, 8t, 10t$ y $12t$.

	Matriz de	$C = 6t$		$C = 8t$		$C = 10t$		$C = 12t$	
Base de datos	**distancias**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**	**GsPSC**	**ExtPSC**
Letter-low	265.96	2.65	4.28	3.21	4.06	3.76	5.23	4.58	4.99
Letter-med	300.67	2.66	13.43	4.59	12.50	4.93	12.11	5.22	10.90
Letter-high	433.37	2.56	15.12	5.39	15.89	6.10	15.17	4.36	15.03
GREC	682.47	1.84	1.74	3.03	2.35	4.02	2.67	2.93	3.00
Fingerprint	1647.27	5.64	49.24	15.55	53.31	23.69	43.82	17.67	46.38
AIDS	4837.12	3.11	13.42	4.02	10.28	3.36	11.35	2.20	7.73
Protein	2702.25	0.40	2.85	0.81	3.43	0.83	2.04	0.53	1.72
Mutagenicity	90361.91	26.81	275.31	22.01	227.41	19.77	188.47	14.42	141.11
Coil-rag	1674.92	37.06	54.90	72.55	63.38	91.01	72.51	73.94	76.70
Coil-del	107106.70	37.09	68.33	53.87	71.95	56.52	81.90	77.89	99.25
Promedio	21001.26	**11.98**	49.86	18.50	46.46	23.36	43.53	22.13	**40.68**

En el caso del algoritmo D-SPS, se hicieron experimentos con diferentes valores para el número de prototipos por clase, de acuerdo a un porcentaje de retención. En la tabla 5.5 se muestran los resultados de clasificación obtenidos con 1-NN, usando valores de retención del 25, 30, 35, 40 y 45 %. Al analizar estos resultados, puede notarse que la clasificación alcanzada con D-SPS es mejor cuando la cantidad de prototipos a seleccionar se acerca al 100 %, lo cual, para fines de la selección de instancias no es recomendable alcanzar este nivel de retención. Con base en estos resultados, en la siguiente sección se utilizó un porcentaje de retención del 45 %, dado que D-SPS obtiene los mejores resultados de clasificación con éste.

Tabla 5.5: Resultados de clasificación con 1-NN para D-SPS, usando porcentajes de retención del 25, 30, 35, 40 y 45 %.

Base de datos	**Orig.**	**25 %**	**30 %**	**35 %**	**40 %**	**45 %**
Letter-low	99.56	98.53	98.80	98.80	98.84	99.07
Letter-med	94.71	76.18	79.11	81.91	83.38	87.56
Letter-high	90.44	64.22	67.69	71.87	75.42	79.47
GREC	99.36	94.73	96.00	96.36	97.00	97.27
Fingerprint	80.54	65.18	65.07	66.61	69.50	63.96
AIDS	99.20	88.00	88.15	89.25	91.00	91.55
Protein	72.33	46.17	47.67	48.00	50.33	54.33
Mutagenicity	70.81	63.22	64.21	64.65	65.81	66.29
Coil-rag	96.90	59.28	63.59	68.08	70.03	74.31
Coil-del	94.69	51.41	57.54	62.21	63.90	67.13
Promedio	89.85	70.69	72.78	74.77	76.52	**78.09**

5.2.3. Comparación experimental

En la tabla 5.6 se muestran los resultados de clasificación y porcentajes de retención obtenidos con los distintos algoritmos para la selección de instancias en el espacio de los grafos, utilizando los clasificadores 1, 3 y 5-NN. Los mejores resultados de clasificación aparecen marcados en negritas. Luego, a partir de los promedios reportados en esta tabla, en las figuras 5.1, 5.2 y 5.3, se muestran las gráficas de comparación entre los algoritmos de selección de instancias presentados.

Con base en estos resultados experimentales, se observa que el algoritmo DROPRel+FSS obtiene en promedio los mejores resultados de clasificación en 1, 3 y 5-NN, seguido por DROPRel. En el caso de D-SPS, éste obtiene en promedio los peores resultados de clasificación, incluso con un nivel de retención mayor al de los demás algoritmos presentados. Por último, en las figuras 5.1, 5.2 y 5.3, es posible observar que en 1 y 3-NN los algoritmos DROPRel, DROPRel+FSS y GsDROP5 forman parte del frente de Pareto, mientras que en 5-NN es modificado por GsDROP3.

Por otra parte, con el propósito de validar los resultados de clasificación, en las figuras 5.4 y 5.5 se muestran los digramas CD, luego de aplicar sobre estos resultados la prueba de *Friedman* y el *post-hoc Bergmman-Hommel*, usando un nivel de significancia $\alpha = 0.05$. Para este propósito, las pruebas estadísticas fueron divididas en dos partes. En la primera parte se evaluaron los algoritmos de tipo *wrapper* como GsDROP3, GsDROP5, DROPRel y DROPRel+FSS. Posteriormente, en la segunda parte se evaluaron los algoritmos de tipo *filter* como GsPSC, ExtPSC y D-SPS, además del mejor algoritmo obtenido en la primera parte. Cabe mencionar que el principal motivo de dividir estas pruebas, es por la recomendación realizada por Demšar (2006), la cual consiste en usar al menos el doble de conjuntos de datos que el número de algoritmos a comparar.

En los diagramas CD de la figura 5.4, se observa que los mejores algoritmos *wrapper* en el ranking son DROPRel+FSS y DROPRel. El análisis estadístico *Bergmman-Hommel* muestra que en 1-NN no existe diferencia significativa entre los algoritmos comparados. Mientras que en 3 y 5-NN, DROPRel+FSS es estadísticamente mejor que GsDROP3 y GsDROP5, pero no así que DROPRel.

Tabla 5.6: Resultados de clasificación en el espacio de los grafos con 1, 3 y 5-NN.

Bases de Datos	**Orig.**		**GsDROP3**		**GsDROP5**		**DROPRel**		**DROPRel+ FSS**		**GsPSC**		**ExtPSC**		**D-SPS**	
	Acc.	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**
1-NN																
Letter-low	99.56	100.00	99.02	1.91	98.84	1.86	99.29	1.87	99.33	1.92	**99.47**	7.98	99.42	7.81	99.07	45.19
Letter-med	94.71	100.00	92.67	9.26	92.27	6.98	**93.20**	8.74	92.58	9.48	88.00	14.94	90.18	18.00	87.56	45.19
Letter-high	90.44	100.00	**89.64**	13.57	87.96	10.55	88.62	12.90	88.67	13.04	83.51	19.60	85.42	25.00	79.47	45.19
GREC	99.36	100.00	99.09	6.56	98.45	6.94	99.09	5.68	99.09	5.68	99.45	22.86	**99.73**	22.62	97.27	46.67
Fingerprint	80.54	100.00	80.79	10.15	81.00	8.38	**81.43**	9.77	80.96	10.88	73.89	23.12	81.11	21.00	63.96	45.08
AIDS	99.20	100.00	**99.45**	1.34	99.25	0.96	99.10	1.42	99.10	1.47	95.30	4.44	99.40	9.00	91.55	45.00
Protein	72.33	100.00	62.17	27.87	61.67	27.70	62.67	28.93	**64.33**	31.91	49.67	21.39	57.17	38.48	54.33	45.56
Mutagenicity	70.81	100.00	70.76	17.23	69.75	14.59	**70.93**	18.50	70.72	21.72	61.63	33.56	68.50	33.44	66.29	45.02
Coil-rag	96.90	100.00	91.72	18.38	92.72	17.72	92.44	17.55	92.51	17.56	**94.10**	34.21	93.38	35.27	74.31	45.87
Coil-del	94.69	100.00	82.44	30.12	82.92	27.94	82.85	30.12	82.90	30.14	87.79	39.46	**88.05**	43.66	67.13	45.87
Promedio	89.85	100.00	86.77	13.64	86.48	12.36	86.96	13.55	**87.02**	14.38	83.28	22.16	86.24	25.43	78.09	45.46
3-NN																
Letter-low	99.51	100.00	99.29	2.38	99.33	2.17	**99.38**	2.50	**99.38**	2.52	99.29	7.98	99.02	7.81	99.07	45.19
Letter-med	95.07	100.00	93.07	9.68	92.49	7.42	**93.73**	10.28	93.20	10.35	90.18	14.94	91.42	15.68	87.91	45.19
Letter-high	92.40	100.00	**90.44**	15.36	89.47	11.88	90.27	15.43	90.27	15.67	86.84	19.60	87.38	27.53	81.91	45.19
GREC	99.55	100.00	99.18	9.79	99.00	9.05	**99.36**	9.20	**99.36**	9.24	98.82	22.86	99.00	22.62	96.91	46.67
Fingerprint	82.29	100.00	82.00	9.51	81.39	10.08	**82.43**	10.99	81.93	12.10	74.32	23.12	81.89	22.31	69.50	45.08
AIDS	99.25	100.00	99.35	1.76	98.95	1.10	99.45	1.87	**99.50**	1.90	97.00	4.44	99.10	11.34	91.40	45.00
Protein	69.17	100.00	52.00	27.65	53.17	29.04	55.33	31.78	**59.00**	35.57	39.33	21.39	46.33	38.48	43.33	45.56
Mutagenicity	72.49	100.00	71.71	17.75	70.99	20.14	70.99	25.04	**71.94**	25.31	63.57	33.56	68.80	33.44	66.08	45.02
Coil-rag	93.62	100.00	88.03	22.60	88.79	21.78	89.41	22.89	**89.79**	27.84	86.36	34.21	87.18	35.27	70.46	45.87
Coil-del	89.23	100.00	74.74	35.45	75.72	32.59	77.97	38.46	**78.10**	38.51	73.10	39.46	75.26	43.66	59.03	45.87
Promedio	89.26	100.00	84.98	15.19	84.93	14.53	85.83	16.84	**86.25**	17.90	80.88	22.16	83.54	25.81	76.56	45.46
5-NN																
Letter-low	99.42	100.00	99.11	2.97	99.11	2.75	99.33	3.31	**99.51**	3.38	99.24	7.98	98.53	7.81	99.11	45.19
Letter-med	95.16	100.00	93.24	10.55	92.58	8.09	**94.18**	11.45	93.11	11.51	91.20	14.94	91.82	15.68	90.27	45.19
Letter-high	93.29	100.00	91.33	17.20	90.27	13.54	**91.38**	17.44	91.16	18.17	88.22	19.60	88.76	27.53	82.53	45.19
GREC	99.55	100.00	98.91	12.47	**99.09**	12.07	**99.09**	12.56	**99.09**	12.65	97.00	22.86	96.09	22.62	96.18	46.67
Fingerprint	83.00	100.00	82.18	9.99	81.61	10.85	82.21	12.14	81.89	13.59	77.75	23.12	**83.29**	22.31	69.68	45.08
AIDS	99.05	100.00	99.40	2.07	99.35	1.18	99.50	2.08	**99.55**	2.12	96.35	4.44	99.05	11.34	91.25	45.00
Protein	62.83	100.00	50.00	24.74	51.33	29.67	52.67	34.22	**58.33**	37.94	34.67	21.39	43.33	38.48	42.00	45.56
Mutagenicity	73.76	100.00	72.58	18.57	71.29	23.61	72.68	29.26	**72.86**	31.59	64.24	33.56	69.15	33.44	65.71	45.02
Coil-rag	91.69	100.00	85.31	25.46	86.97	26.15	88.18	27.80	**88.38**	28.83	82.49	34.21	82.46	35.27	68.08	45.87
Coil-del	85.10	100.00	68.56	37.32	72.95	37.92	75.54	44.48	**75.69**	44.54	66.82	39.46	68.33	43.66	54.51	45.87
Promedio	88.29	100.00	84.06	16.13	84.46	16.58	85.48	19.47	**85.96**	20.43	79.80	22.16	82.08	25.81	75.93	45.46

Los diagramas CD de los algoritmos de tipo *filter* y DROPRel+FSS el mejor en la primera prueba estadística, se muestran en la figura 5.5. Con base en estos resultados, es posible observar que DROPRel+FSS es el algoritmo mejor situado en el ranking para los clasificadores 3 y 5-NN, pero no así en 1-NN donde el mejor ubicado es ExtPSC. Este último algoritmo, es el mejor de tipo *filter* situado en el ranking y sus resultados muestran que para 1 y 3-NN, no hay una diferencia significativa en los resultados obtenidos con respecto a DROPRel+FSS.

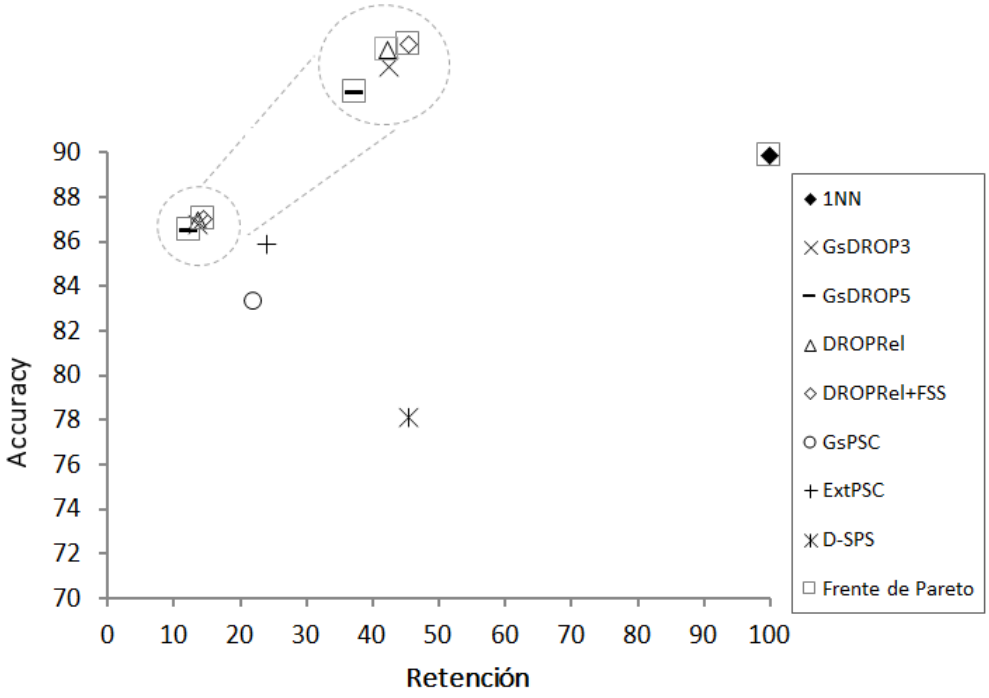

Figura 5.1: Comparación de los algoritmos de selección de instancias en el espacio de los grafos, usando el clasificador 1-NN.

En la tabla 5.7 por otro lado, se muestran los tiempos de ejecución promedio para cada algoritmo de selección de instancias presentado. Estos tiempos fueron tomados posterior al cálculo de la matriz de distancias entre grafos en cada base de datos.

Con base en los resultados de la tabla 5.7, puede observarse que GsPSC obtiene en promedio el mejor tiempo, seguido por ExtPSC que en términos de *accuracy* fue el mejor algoritmo *filter*. Por otro lado, DROPRel+FSS y D-SPS obtienen los peores resultados en tiempo de ejecución promedio.

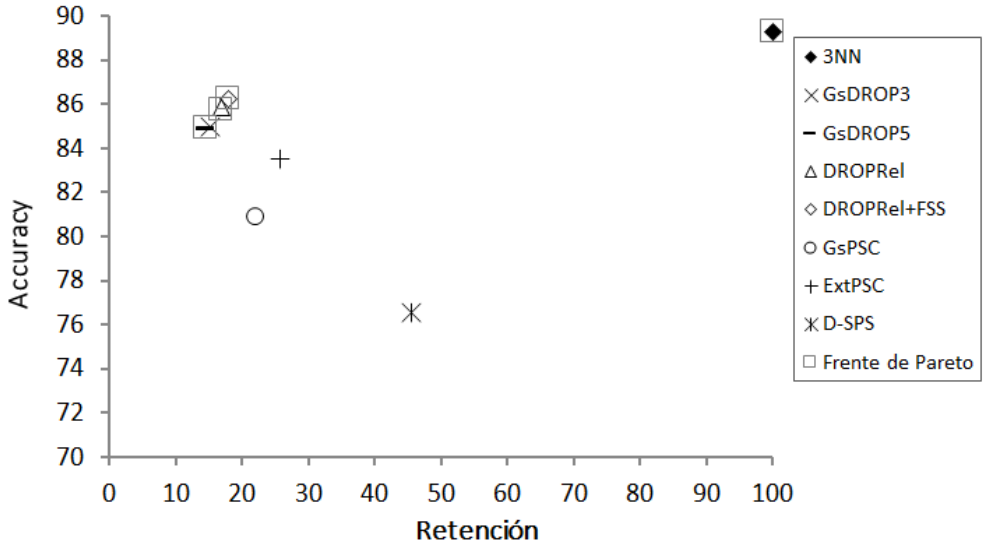

Figura 5.2: Comparación de los algoritmos de selección de instancias en el espacio de los grafos, usando el clasificador 3-NN.

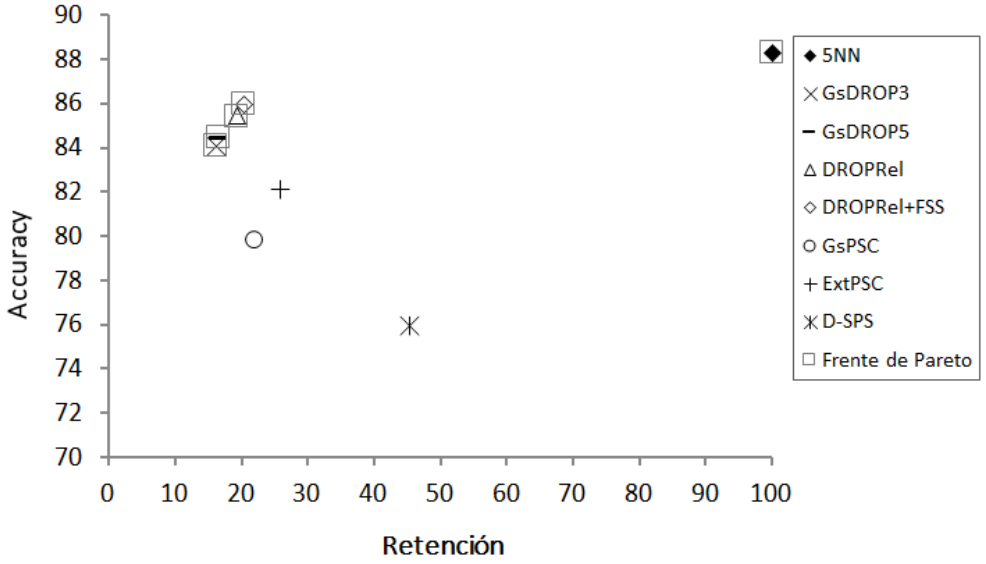

Figura 5.3: Comparación de los algoritmos de selección de instancias en el espacio de los grafos, usando el clasificador 5-NN.

De acuerdo con lo descrito en esta sección, el algoritmo de tipo *wrapper* DROPRel+FSS obtiene en promedio los mejores resultados de clasificación para 1, 3 y 5-NN, aunque requiere mayores tiempos de ejecución. Por otro lado, ExtPSC es

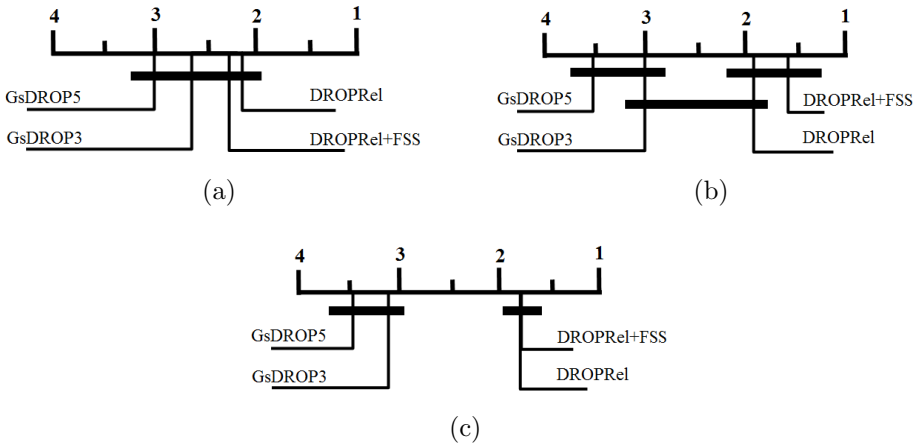

Figura 5.4: Diagramas CD para los algoritmos wrapper: (a) 1-NN, b) 3-NN y (c) 5-NN.

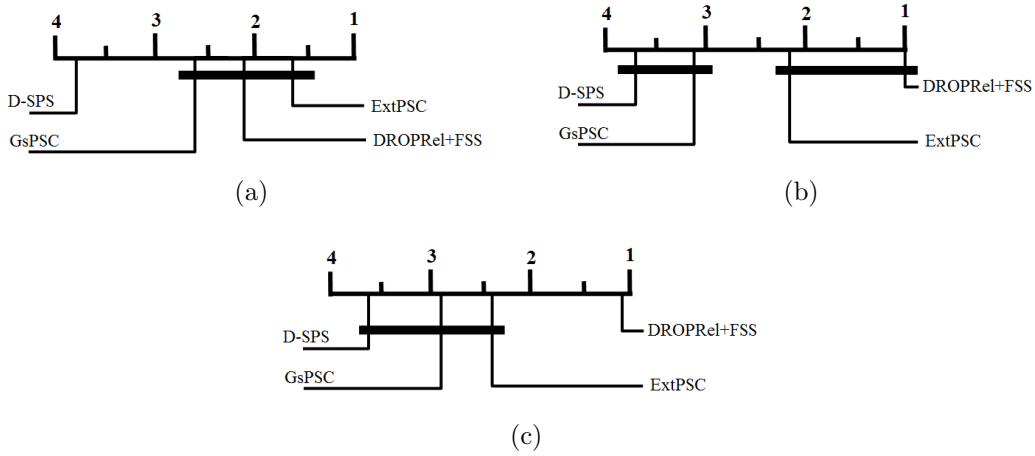

Figura 5.5: Diagramas CD para los algoritmos filter y DROPRel+FSS: (a) 1-NN, b) 3-NN y (c) 5-NN.

el algoritmo de tipo *filter* con los mejores resultados de clasificación, logrando incluso resultados estadísticamente similares a los alcanzados con DROPRel+FSS en 1 y 3-NN. En lo que respecta al tiempo de ejecución promedio, ExtPSC obtiene los segundos mejores resultados, después de GsPSC, que aunque este último es el más rápido, en términos de la calidad de clasificación no es una buena opción.

Tabla 5.7: Tiempo por calcular la matriz de distancias entre grafos para cada base de datos y tiempos de ejecución promedio para los algoritmos de selección de instancias en el espacio de los grafos. Los tiempos reportados en esta tabla están dados en segundos.

Base de Datos	Matriz de distancias	GsDROP3	GsDROP5	DROPRel	DROPRel+ FSS	GsPSC	ExtPSC	D-SPS
Letter-low	265.96	2.16	0.93	0.93	148.68	3.76	5.23	776.16
Letter-med	300.67	1.08	0.69	0.61	643.65	4.93	12.11	781.62
Letter-high	433.37	0.91	0.60	0.67	860.74	6.10	15.17	760.47
GREC	682.47	0.18	0.13	0.15	48.77	4.02	2.67	55.77
Fingerprint	1647.27	707.09	501.14	595.09	1870.37	23.69	43.82	5356.88
AIDS	4837.12	1.42	1.01	1.19	143.43	3.36	11.35	2468.12
Protein	2702.25	0.02	0.01	0.02	49.63	0.83	2.04	27.78
Mutagenicity	90361.91	3.90	2.32	4.22	14780.14	19.77	188.47	33844.00
Coil-rag	1674.92	3.63	2.38	3.23	7251.11	91.01	72.51	1149.87
Coil-del	107106.70	2.60	2.12	2.57	9590.54	56.52	81.90	897.58
Promedio	21001.26	72.30	51.13	60.87	3538.71	**21.40**	43.53	4611.83

5.3. Resultados experimentales para la selección de instancias en el espacio n-dimensional

En esta sección se presentan los resultados obtenidos para los algoritmos de tipo *wrapper* DsDROPRel y DsDROPRel+FSS, además del algoritmo de tipo *filter* DsExtPSC, todos ellos propuestos en el espacio de representación n-dimensional generado por la técnica *embedding de grafos vía disimilaridad*. Primero, se describe la configuración utilizada en esta técnica para generar el nuevo espacio de representación de los grafos. Luego, se muestran los algoritmos de referencia utilizados en nuestra comparación. En seguida, se explican los parámetros usados por los distintos algoritmos de selección de instancias. Finalmente, se presentan los resultados obtenidos en la comparación experimental.

5.3.1. Generación del espacio de n-dimensional

Con el propósito de generar el espacio de representación n-dimensional mediante el *embedding de grafos vía disimilaridad*, todos los elementos en el conjunto de entrenamiento fueron usados como conjunto de prototipos (ver sección 2.4). No obstante, cualquier algoritmo de *selección de prototipos para embedding de grafos* (Borzeshi et al., 2013; Livi et al., 2014; Riesen and Bunke, 2009b,c; Riesen et al., 2007) puede ser utilizado con el fin de reducir la dimensionalidad de los vectores n-dimensionales

de características generados. La razón de utilizar todo el conjunto de entrenamiento como conjunto de prototipos en nuestros experimentos, es para evitar la pérdida de información que se pudiera ocasionar al reducir la dimensionalidad de estos vectores de características.

5.3.2. Algoritmos usados para la comparación

En el espacio de representación n-dimensional se realiza una comparación experimental con respecto a los algoritmos de tipo *wrapper* como DROP3 y DROP5 (Wilson and Martinez, 2000), además del algoritmo de tipo *filter* PSC (Olvera-López et al., 2010). Estos algoritmos fueron elegidos para nuestra comparación ya que son de los más exitosos en términos de calidad de clasificación, reducción y tiempo de ejecución reportados en la literatura.

5.3.3. Ajuste de parámetros

Los algoritmos *wrapper* como DROP3, DROP5, DsDROPRel y DsDROPRel+FSS, no necesitan de la configuración de algún parámetro para su ejecución. Mientras que algoritmos *filter* como PSC y DsExtPSC usan como parámetro el número de grupos C a generar. De la misma manera que en el espacio de los grafos, se hicieron experimentos con diferentes valores para determinar el parámetro C. En las tablas 5.8 y 5.9, se muestran los resultados de clasificación y retención obtenidos con 1-NN, utilizando los valores $C = 6t, 8t, 10t$ y $12t$ (donde t es nuevamente el número de clases).

Basados en las tablas 5.8 y 5.9, los mejores resultados de clasificación promedio son obtenidos con $C = 12t$. Sin embargo, el valor seleccionado fue $C = 10t$, ya que éste obtiene una calidad de clasificación similar a la de $C = 12t$ pero con un mejor porcentaje de retención.

5.3.4. Comparación experimental

En la tabla 5.10 se muestran los resultados de clasificación y de retención obtenidos con los algoritmos de selección de instancias presentados en esta sección, utilizando k-NN con k=1, 3 y 5. Los mejores resultados de clasificación se indican en

Tabla 5.8: Resultados de clasificación obtenidos con PSC y DsExtPSC, usando $C = 6t, 8t, 10t$ y $12t$.

Dataset	**Orig.**	**Número de grupos**							
		$C = 6t$		$C = 8t$		$C = 10t$		$C = 12t$	
		PSC	DsExtPSC	PSC	DsExtPSC	PSC	DsExtPSC	PSC	DsExtPSC
Letter-low	99.33	98.22	98.80	98.71	98.98	98.80	98.89	98.76	99.07
Letter-med	93.29	85.24	87.91	87.02	90.18	88.93	90.31	88.53	90.04
Letter-high	90.44	81.24	82.22	81.42	84.31	82.31	85.20	83.16	84.53
GREC	99.09	98.55	98.00	98.00	98.64	98.45	98.45	98.64	98.45
Fingerprint	79.79	68.39	75.89	70.00	76.57	69.93	77.68	70.54	76.75
AIDS	99.50	94.40	98.95	94.85	98.70	95.00	99.25	95.45	99.40
Protein	64.50	38.50	51.33	37.50	48.83	44.17	47.83	43.17	52.17
Mutagenicity	71.32	61.19	66.33	60.59	66.10	60.94	67.10	60.64	67.17
Coil-rag	94.44	90.62	90.23	91.05	90.90	91.69	91.69	92.56	91.69
Coil-del	84.33	74.08	75.18	76.67	76.46	77.56	77.18	79.26	77.90
Promedio	87.60	79.04	82.49	79.58	82.97	80.78	83.36	**81.07**	**83.72**

Tabla 5.9: Resultados de retención obtenidos con PSC y DsExtPSC, usando $C = 6t, 8t, 10t$ y $12t$.

Dataset	**Orig.**	**Número de grupos**							
		$C = 6t$		$C = 8t$		$C = 10t$		$C = 12t$	
		PSC	DsExtPSC	PSC	DsExtPSC	PSC	DsExtPSC	PSC	DsExtPSC
Letter-low	100.00	5.67	6.00	6.86	7.16	8.52	8.58	9.99	9.64
Letter-med	100.00	13.12	14.95	14.16	15.05	15.10	15.97	16.45	17.45
Letter-high	100.00	19.72	20.71	19.64	21.99	19.57	22.32	20.76	21.83
GREC	100.00	15.31	15.69	19.84	19.99	23.46	23.07	27.59	27.31
Fingerprint	100.00	16.22	18.19	16.69	18.65	16.68	19.00	17.29	19.19
AIDS	100.00	2.21	2.29	2.06	2.08	2.21	2.24	2.39	2.62
Protein	100.00	29.37	41.44	30.78	42.74	33.02	44.11	34.15	46.52
Mutagenicity	100.00	37.24	31.08	37.08	31.06	37.04	31.22	36.66	31.49
Coil-rag	100.00	26.44	27.75	30.38	30.92	34.85	34.81	39.22	39.43
Coil-del	100.00	39.57	46.38	42.59	46.83	44.84	47.72	48.51	49.81
Promedio	100.00	**20.49**	**22.45**	22.01	23.65	23.53	24.91	25.30	26.53

negritas. Con base en los promedios reportados en estas tablas, en las figuras 5.6, 5.7 y 5.8 se presentan gráficas de comparación para los algoritmos presentados.

De acuerdo a estos resultados experimentales, se observa que el algoritmo DsDROPRel+FSS obtiene los mejores resultados de clasificación para el espacio de representación n-dimensional con los clasificadores 1, 3 y 5-NN. En las figuras 5.6, 5.7 y 5.8, es posible observar que DsDROPRel+FSS y DROP5 constituyen el frente de Pareto en 1-NN; mientras que en los clasificadores 3 y 5-NN, el frente Pareto es modificado por DROP3 y DsDROPRel.

Tabla 5.10: Resultados de clasificación en el espacio n-dimensional obtenido mediante el embedding de grafos, con 1, 3 y 5-NN.

Base de Datos	**Orig.**		**DROP3**		**DROP5**		**DsDROPRel**		**DsDROPRel+ FSS**		**PSC**		**DsExtPSC**	
	Acc.	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**
1-NN														
Letter-low	99.29	100.00	98.44	3.36	98.67	2.55	98.58	2.42	98.76	2.84	98.80	8.52	**98.89**	8.58
Letter-med	93.96	100.00	90.67	9.69	90.53	8.79	**91.38**	9.39	91.29	11.64	88.93	15.10	90.31	15.97
Letter-high	90.31	100.00	87.16	14.04	87.07	11.91	**87.78**	13.43	87.51	16.22	82.31	19.57	85.20	22.32
GREC	99.27	100.00	98.09	10.09	98.09	10.09	98.36	8.68	98.27	9.10	**98.45**	23.46	**98.45**	23.07
Fingerprint	79.54	100.00	78.29	8.41	77.43	9.35	76.32	8.92	**80.75**	13.11	69.93	16.68	77.68	19.00
AIDS	99.15	100.00	**99.45**	1.03	98.85	1.24	99.25	0.87	**99.45**	1.02	95.00	2.21	99.25	2.24
Protein	65.33	100.00	49.50	33.80	53.33	35.41	54.83	35.74	**58.50**	50.83	44.17	33.02	47.83	44.11
Mutagenicity	71.80	100.00	68.11	21.91	69.06	21.84	**69.40**	23.97	**69.40**	30.49	60.94	37.04	67.10	31.22
Coil-rag	94.51	100.00	88.46	21.59	88.31	20.38	88.77	20.99	91.26	24.30	**91.69**	34.85	**91.69**	34.81
Coil-del	84.13	100.00	69.44	33.46	70.28	32.68	70.41	33.40	76.33	41.04	**77.56**	44.84	77.18	47.72
Promedio	87.73	100.00	82.76	15.74	83.16	15.42	83.51	15.78	**85.15**	20.06	80.78	23.53	83.36	24.91
3-NN														
Letter-low	99.16	100.00	98.44	4.62	97.47	3.71	98.40	3.73	**98.62**	4.12	97.91	8.52	**98.62**	8.58
Letter-med	93.91	100.00	92.18	10.67	91.73	10.16	91.78	11.00	**92.27**	12.92	88.98	15.10	91.38	15.97
Letter-high	91.60	100.00	88.93	16.89	87.96	14.18	89.07	16.43	**89.24**	18.95	83.29	19.57	85.51	22.32
GREC	98.64	100.00	**97.36**	13.79	**97.36**	13.79	96.91	12.99	97.18	13.55	93.18	23.46	92.18	23.07
Fingerprint	81.93	100.00	79.25	7.96	75.57	11.00	72.46	10.46	**81.18**	12.86	69.71	16.68	78.25	19.00
AIDS	55.50	100.00	99.35	1.38	98.85	1.60	99.10	1.42	**99.45**	1.60	97.10	2.21	99.10	2.24
Protein	99.65	100.00	44.83	31.85	45.33	40.81	47.17	41.11	**52.50**	49.48	31.67	33.02	39.17	44.11
Mutagenicity	71.62	100.00	**70.28**	20.78	69.43	24.50	69.56	27.93	69.82	31.18	57.55	37.04	65.02	31.22
Coil-rag	90.67	100.00	84.03	25.64	85.05	25.34	85.79	27.06	**87.69**	29.17	81.18	34.85	81.97	34.81
Coil-del	76.64	100.00	62.90	35.07	64.15	37.83	65.44	40.70	**69.95**	48.07	64.21	44.84	65.03	47.72
Promedio	85.93	100.00	81.76	16.86	81.29	18.29	81.57	19.28	**83.79**	22.19	76.48	23.53	79.62	24.91
5-NN														
Letter-low	99.16	100.00	98.40	5.75	98.09	5.08	98.62	4.95	**98.71**	5.28	96.89	8.52	97.82	8.58
Letter-med	93.87	100.00	**92.44**	11.69	92.00	11.20	92.40	13.00	92.31	14.49	89.56	15.10	90.67	15.97
Letter-high	92.27	100.00	88.49	19.39	88.00	15.84	88.67	19.79	**89.16**	21.72	84.80	19.57	86.27	22.32
GREC	98.36	100.00	**96.73**	17.42	**96.73**	17.42	96.36	16.72	97.09	17.27	89.00	23.46	91.09	23.07
Fingerprint	82.96	100.00	81.00	8.38	77.29	11.43	80.14	10.96	**82.14**	13.13	72.04	16.68	78.68	19.00
AIDS	99.65	100.00	99.20	1.68	98.85	1.93	99.25	1.82	**99.30**	1.94	95.85	2.21	98.30	2.24
Protein	51.33	100.00	41.00	30.65	**46.67**	46.96	40.67	46.39	45.33	54.50	29.67	33.02	33.33	44.11
Mutagenicity	71.15	100.00	70.07	20.47	**70.95**	25.79	69.84	30.83	69.89	33.46	59.14	37.04	65.69	31.22
Coil-rag	88.64	100.00	81.51	28.42	83.82	29.78	83.77	32.11	**85.92**	34.33	75.69	34.85	76.18	34.81
Coil-del	73.21	100.00	58.97	35.61	63.62	43.09	63.67	46.09	**66.72**	52.59	60.51	44.84	61.21	47.72
Promedio	85.06	100.00	80.78	17.95	81.60	20.85	81.34	22.27	**82.66**	24.87	75.31	23.53	77.92	24.91

Para validar los resultados de clasificación, en las figuras 5.9 y 5.10 se muestran los diagramas CD, después de aplicar sobre estos resultados la prueba de *Friedman* y el *post-hoc Bergmman-Hommel*, usando un nivel de significancia $\alpha = 0.05$. Al igual que en el espacio de los grafos, estas pruebas estadísticas fueron divididas en dos partes. En la primera parte se evaluaron algoritmos de tipo *wrapper* como DROP3, DROP5, DsDROPRel y DsDROPRel+FSS. Mientras que en la segunda parte se evaluaron los algoritmos *filter* DsExtPSC, PSC y la mejor solución de la primera evaluación.

Los diagramas CD de la figura 5.9 muestran que el mejor algoritmo en el ranking es DsDROPRel+FSS. El *post-hoc Bergmann-Hommel* indica que este algoritmo obtiene una mejora significativa con respecto a DROP3, DROP5 y DsDROPRel, usando el clasificador 1-NN. En el caso de 3-NN, DsDROPRel+FSS es estadísticamente mejor que DROP5 y DsDROPRel, pero no que DROP3. Por último, para el clasificador 5-NN todos los algoritmos comparados son estadísticamente similares.

En la figura 5.10 por otro lado, se muestran los diagramas CD para los algoritmos PSC y DsExtPSC, además de DsDROPRel+FSS el mejor del primer análisis estadístico. Con base en estos diagramas, se observa nuevamente que DsDROPRel+FSS es el mejor algoritmo en el ranking para todos los clasificadores utilizados, siendo estadísticamente mejor que PSC y ExtPSC en 3 y 5-NN.

En la tabla 5.11 por otro lado, se muestran los tiempos de ejecución del *embedding de grafos vía disimilaridad*, así como los tiempos de ejecución promedio de cada algoritmo de selección de instancias para el espacio de representación n-dimensional. Con base en los resultados de esta tabla, DsDROPRel+FSS y DROP5 son los algoritmos con el mayor tiempo de ejecución promedio. Mientras que PSC obtiene los menores tiempos de ejecución, seguido por el algoritmo DsExtPSC.

De acuerdo a lo reportado en esta sección, puede observarse que el algoritmo *wrapper* DsDROPRel+FSS obtiene los mejores resultados de clasificación en 1, 3 y 5-NN; sin embargo, el tiempo de ejecución de éste es mayor al del resto de los algoritmos. Por otro lado, DsExtPSC es el algoritmo *filter* con los mejores resultados de clasificación, incluso con resultados estadísticamente similares a los

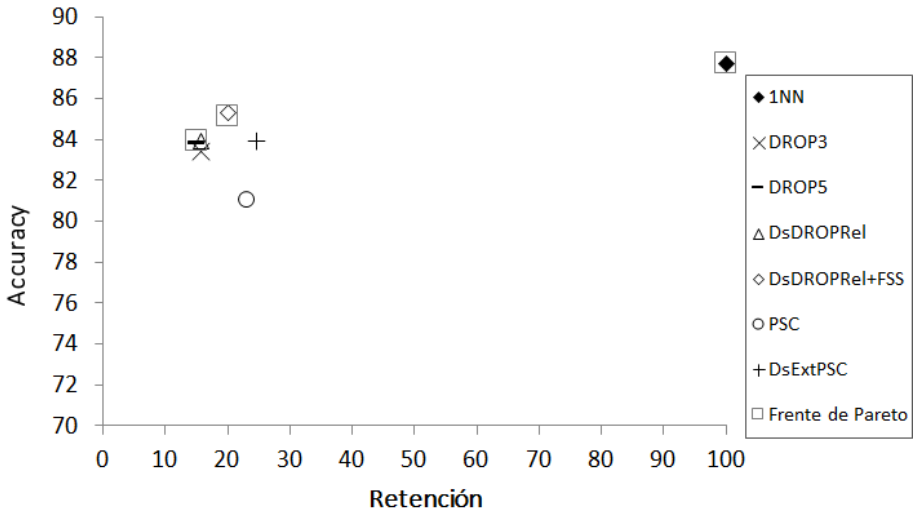

Figura 5.6: Comparación de los algoritmos de selección de instancias en el espacio n-dimensional, usando el clasificador 1-NN.

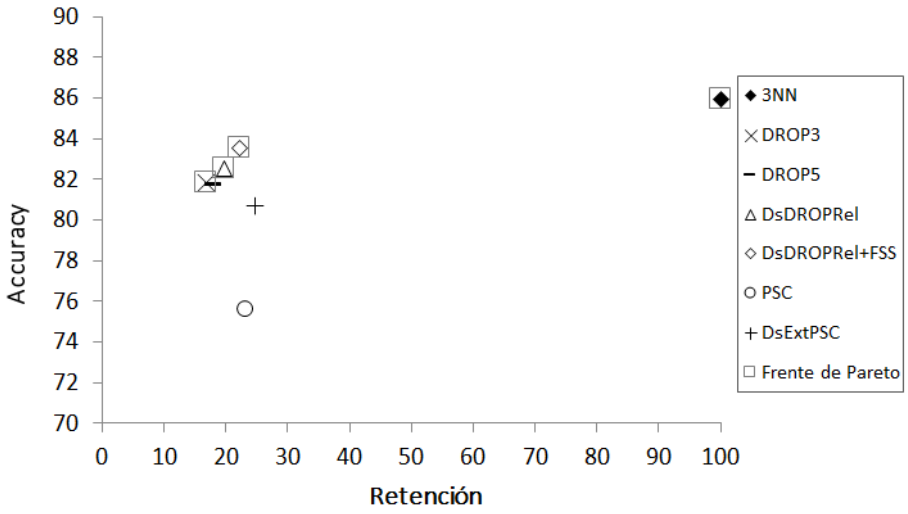

Figura 5.7: Comparación de los algoritmos de selección de instancias en el espacio n-dimensional, usando el clasificador 3-NN.

de DsDROPRel+FSS en 1-NN. El tiempo de DsExtPSC es en promedio el segundo mejor de todos los algoritmos presentados, después de PSC.

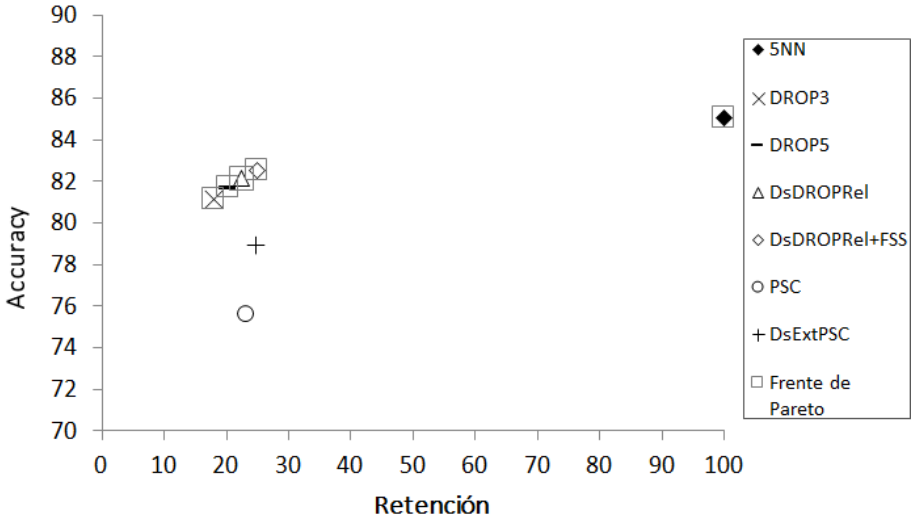

Figura 5.8: Comparación de los algoritmos de selección de instancias en el espacio n-dimensional, usando el clasificador 5-NN.

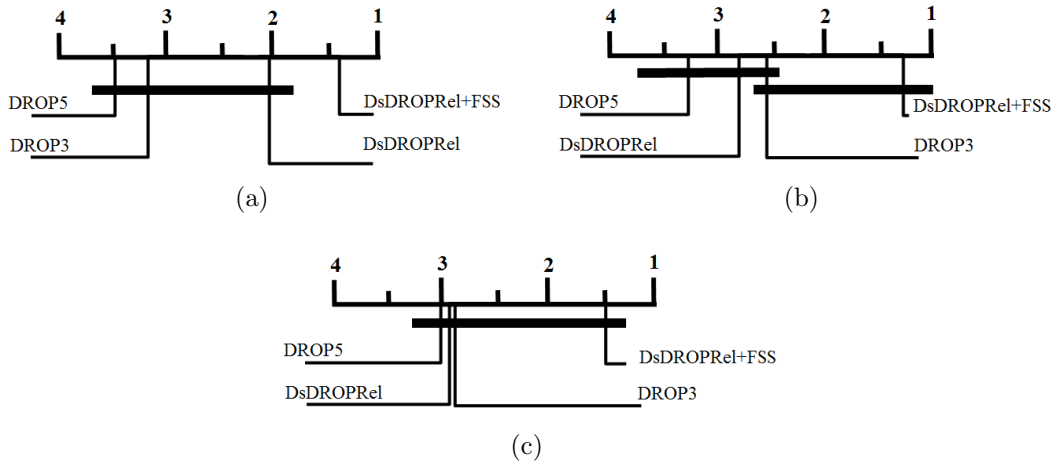

Figura 5.9: Diagramas CD para los algoritmos wrapper: (a) 1-NN, b) 3-NN y (c) 5-NN.

5.4. Resultados experimentales para la selección de instancias híbrida

En esta sección se presentan los resultados obtenidos para la selección de instancias híbrida correspondiente a los algoritmos HDroprel+FSS e HExtPSC+FSS. Primero, se muestra la configuración y parámetros utilizados por cada algoritmo. Luego, se presenta una comparación entre estas soluciones. Cabe señalar que aunque algoritmos

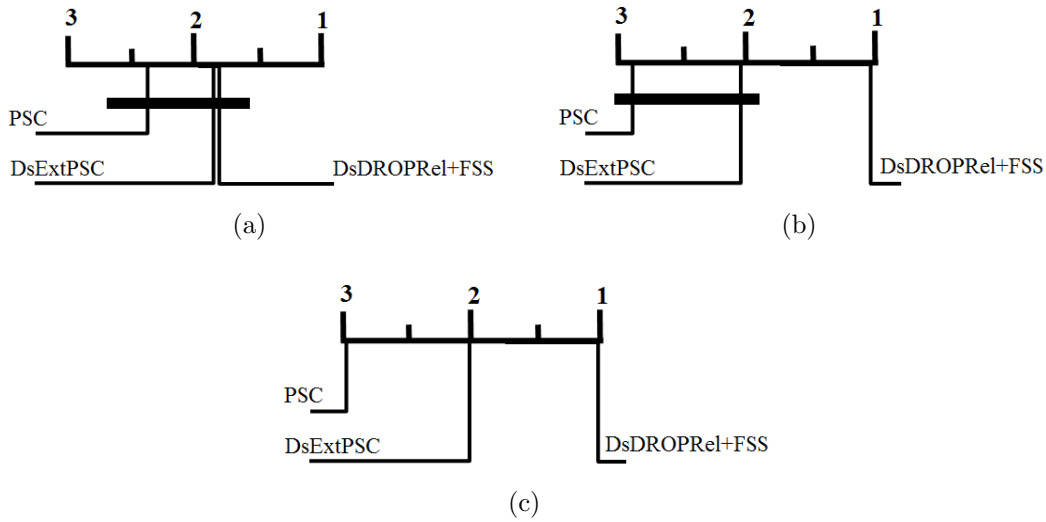

Figura 5.10: Diagramas CD para los algoritmos filter y DsDROPRel+FSS: (a) 1-NN, b) 3-NN y (c) 5-NN.

Tabla 5.11: Tiempo del embedding de grafos vía disimilaridad y tiempo de ejecución promedio de los algoritmos de selección de instancias en el espacio n-dimensional (segundos).

Base de Datos	**Tiempo Embedding**	**DROP3**	**DROP5**	**DsDROPRel**	**DsDROPRel +FSS**	**PSC**	**DsExtPSC**
Letter-low	265.96	3607.09	4557.28	1944.59	2854.57	138.18	190.30
Letter-med	300.67	2242.72	2761.81	1296.93	5373.75	145.38	221.78
Letter-high	433.37	1862.16	2798.47	1223.43	8194.38	99.42	232.53
GREC	682.47	300.18	310.78	201.45	578.13	28.01	24.61
Fingerprint	1647.27	75862.39	197492.83	48895.71	54721.41	163.39	476.82
AIDS	4837.12	5276.67	2715.57	3766.41	4266.41	43.57	76.58
Protein	2702.25	16.17	14.12	20.17	269.88	5.34	6.27
Mutagenicity	90361.91	8096.32	7652.27	10932.00	73511.38	739.54	6621.12
Coil-rag	1674.92	5456.76	6636.85	9455.20	33321.20	2596.30	2286.29
Coil-del	107106.70	5991.18	4505.86	3583.64	49227.77	1743.84	2916.14
Promedio	21001.26	10871.16	22944.58	8131.95	23231.89	**570.30**	1305.24

de tipo híbrido no se reportan en la literatura, el espacio de representación en el que es utilizado el conjunto de entrenamiento reducido es el n-dimensional; por lo que algoritmos de referencia en este espacio de representación pudieran ser utilizados para nuestra comparación. En la sección 5.5 comparamos nuestro mejor algoritmo híbrido con respecto a la mejor solución del espacio n-dimensional.

5.4.1. Ajuste de parámetros

El algoritmo HDroprel+FSS no requiere de algún parámetro para su ejecución. Mientras que HExtPSC+FSS usa el número de grupos C a generar. Este parámetro C corresponde a la configuración del algoritmo ExtPSC aplicado en el espacio de los grafos. Para determinar el valor de C, atendiendo al estudio experimental mostrado en la tabla 5.2 de la sección 5.2.2, el cual muestra los resultados de clasificación con $C = 6t$, $8t$, $10t$ y $12t$, seleccionamos $C = 10t$, por lo que este valor fue utilizado para el algoritmo HExtPSC+FSS.

Tabla 5.12: Resultados de clasificación para los algoritmos de selección híbridos con 1, 3 y 5-NN.

Base de datos	**Orig.**		**HDroprel +FSS**		**HExtPSC +FSS**	
	Acc.	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**
1-NN						
Letter-low	99.29	100.00	98.40	3.70	**98.53**	8.90
Letter-med	93.96	100.00	**92.40**	14.92	92.22	21.99
Letter-high	90.31	100.00	87.69	20.12	**87.78**	32.33
GREC	99.27	100.00	97.73	8.37	**98.82**	23.69
Fingerprint	79.54	100.00	**80.48**	15.71	79.86	28.47
AIDS	99.15	100.00	98.95	2.83	**99.45**	3.96
Protein	65.33	100.00	53.17	39.87	**54.83**	52.33
Mutagenicity	71.80	100.00	**69.89**	30.85	69.72	45.90
Coil-rag	94.51	100.00	91.00	24.07	**92.54**	38.68
Coil-del	84.13	100.00	77.10	44.75	**79.33**	51.93
Promedio	87.73	100.00	84.68	20.52	**85.31**	30.82
3-NN						
Letter-low	99.16	100.00	98.31	3.92	**99.02**	8.54
Letter-med	93.91	100.00	91.82	14.99	**92.22**	20.92
Letter-high	91.60	100.00	87.78	21.31	**88.89**	30.92
GREC	98.64	100.00	90.55	10.81	**98.55**	24.36
Fingerprint	81.93	100.00	80.48	14.79	**81.18**	25.65
AIDS	55.50	100.00	**99.60**	2.47	99.40	3.87
Protein	99.65	100.00	43.83	40.56	**47.17**	49.22
Mutagenicity	71.62	100.00	68.76	33.18	**69.68**	41.63
Coil-rag	90.67	100.00	83.64	27.50	**87.87**	40.71
Coil-del	76.64	100.00	67.85	51.59	**70.31**	55.58
Promedio	85.93	100.00	81.26	22.11	**83.43**	30.14
5-NN						
Letter-low	99.16	100.00	98.31	4.05	**99.24**	8.66
Letter-med	93.87	100.00	91.69	15.75	**92.58**	20.33
Letter-high	92.27	100.00	**89.20**	23.20	88.98	30.85
GREC	98.36	100.00	90.27	14.26	**95.73**	24.57
Fingerprint	82.96	100.00	80.48	15.26	**82.11**	24.90
AIDS	99.65	100.00	99.20	2.59	**99.50**	3.90
Protein	51.33	100.00	42.50	41.31	**45.00**	48.43
Mutagenicity	71.15	100.00	**69.82**	35.98	68.39	40.26
Coil-rag	88.64	100.00	80.26	32.06	**84.38**	41.21
Coil-del	73.21	100.00	64.72	56.00	**65.97**	55.52
Promedio	85.06	100.00	80.64	24.05	**82.19**	29.86

5.4.2. Comparación experimental

Los resultados de clasificación y retención obtenidos por los algoritmos de selección de instancias híbridos se muestran en la tabla 5.12, usando los clasificadores 1, 3 y 5-NN. Los mejores resultados de clasificación se resaltan en negritas. Con base en estos resultados, el algoritmo HExtPSC+FSS obtiene los mejores resultados de clasificación en todos los clasificadores. Mas aún, un análisis estadístico mediante el *post-hoc Bergmman-Hommel* con un nivel de significancia $\alpha = 0.05$, muestra que HExtPSC+FSS es estadísticamente mejor que HDroprel+FSS usando 3-NN, mientras que en 1 y 5-NN los resultados de clasificación de ambos algoritmos son estadísticamente similares.

En la tabla 5.13 por otro lado, se muestran los tiempos para calcular la matriz de distancias, así como los tiempos de ejecución promedio de los algoritmos híbridos. Con base en estos resultados, podemos observar que HDroprel+FSS obtiene los mejores tiempos de ejecución promedio.

Tabla 5.13: Tiempo de ejecución para los algoritmos de selección de instancias híbrido.

Base de datos	**Matriz de distancias**	**HDroprel+FSS**	**HExtPSC+FSS**
Letter-low	265.96	171.77	565.21
Letter-med	300.67	2022.63	3888.93
Letter-high	433.37	3567.92	6786.12
GREC	682.47	61.23	179.35
Fingerprint	1647.27	3030.27	5669.52
AIDS	4837.12	51.73	163.68
Protein	2702.25	34.99	51.75
Mutagenicity	90361.91	49914.80	60767.85
Coil-rag	1674.92	12873.68	21838.51
Coil-del	108106.70	38573.85	43626.03
Promedio	21101.26	**11030.29**	14353.695

5.5. Comparación de las soluciones propuestas

En esta sección se comparan los algoritmos propuestos en los diferentes enfoques de selección de instancias para bases de datos de grafos; específicamente, los algoritmos de tipo *wrapper* DROPRel, DROPRel+FSS, DsDROPRel y DsDROPRel+FSS, además de los algoritmos de tipo *filter* ExtPSC y DsExtPSC. La tabla 5.14 muestra

un resumen con los resultados de clasificación y retención obtenidos. Los mejores resultados de clasificación se indican nuevamente en negritas. A partir de los promedios reportados en esta tabla, en las figuras 5.11, 5.12 y 5.13 se presentan las gráficas de comparación entre los distintos algoritmos de selección de instancias propuestos.

Con base en estos resultados, se observa que el algoritmo DROPRel+FSS obtiene la mejor calidad de clasificación en 1, 3 y 5-NN, seguido por DROPRel. El algoritmo ExtPSC por otro lado, obtiene resultados de clasificación superiores a soluciones como DsExtPSC y DsDROPRel, mientras que sus resultados son similares a los de DsDROPRel+FSS e HExtPSC+FSS (las mejores soluciones en el espacio n-dimensional e híbrida respectivamente). En las figuras 5.11, 5.12 y 5.13, es posible observar que los algoritmos DROPRel y DROPRel+FSS se encuentran en el frente de Pareto, ambos algoritmos propuestos en el espacio de representación de los grafos.

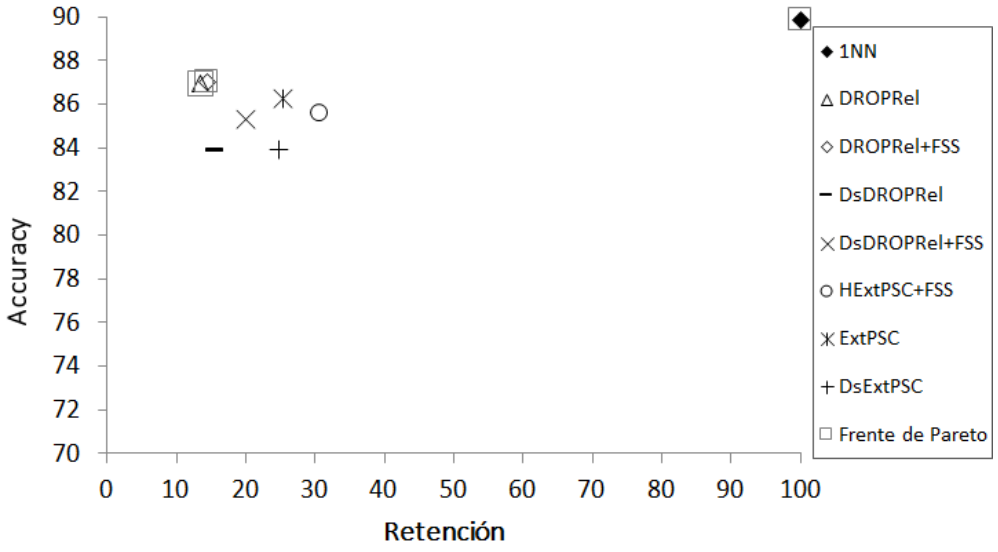

Figura 5.11: Comparación de las soluciones propuestas en los diferentes enfoques de selección, usando el clasificador 1-NN.

Tabla 5.14: Resultados de clasificación para los distintos enfoques de selección de instancias en bases de datos de grafos con 1, 3 y 5-NN.

Base de Datos	**Orig.**		**DROPRel**		**DROPRel+ FSS**		**DsDROPRel**		**DsDROPRel+ FSS**		**HExtPSC+FSS**		**ExtPSC**		**DsExtPSC**	
	Acc.	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**	**Acc.**	**Ret.**
1-NN																
Letter-low	99.56	100.00	99.29	1.87	99.33	1.92	98.58	2.42	98.76	2.84	98.53	8.90	**99.42**	7.81	98.89	8.58
Letter-med	94.71	100.00	**93.20**	8.74	92.58	9.48	91.38	9.39	91.29	11.64	92.22	21.99	90.18	18.00	90.31	15.97
Letter-high	90.44	100.00	88.62	12.90	**88.67**	13.04	87.78	13.43	87.51	16.22	87.78	32.33	85.42	25.00	85.20	22.32
GREC	99.36	100.00	99.09	5.68	99.09	5.68	98.36	8.68	98.27	9.10	98.82	23.69	**99.73**	22.62	98.45	23.07
Fingerprint	80.54	100.00	**81.43**	9.77	80.96	10.88	76.32	8.92	80.75	13.11	79.86	28.47	81.11	21.00	77.68	19.00
AIDS	99.20	100.00	99.10	1.42	99.10	1.47	99.25	0.87	**99.45**	1.02	**99.45**	3.96	99.40	9.00	99.25	2.24
Protein	72.33	100.00	62.67	28.93	**64.33**	31.91	54.83	35.74	58.50	50.83	54.83	52.33	57.17	38.48	47.83	44.11
Mutagenicity	70.81	100.00	**70.93**	18.50	70.72	21.72	69.40	23.97	69.40	30.49	69.72	45.90	68.50	33.44	67.10	31.22
Coil-rag	96.90	100.00	92.44	17.55	92.51	17.56	88.77	20.99	91.26	24.30	92.54	38.68	**93.38**	35.27	91.69	34.81
Coil-del	94.69	100.00	82.85	30.12	82.90	30.14	70.41	33.40	76.33	41.04	79.33	51.93	**88.05**	43.66	77.18	47.72
Promedio	89.85	100.00	86.96	13.55	**87.02**	14.38	83.51	15.78	85.15	20.06	85.31	30.82	86.24	25.43	83.36	24.91
3-NN																
Letter-low	99.51	100.00	**99.38**	2.50	**99.38**	2.52	98.40	3.73	98.62	4.12	99.02	8.54	99.02	7.81	98.62	8.58
Letter-med	95.07	100.00	**93.73**	10.28	93.20	10.30	91.78	11.00	92.27	12.92	92.22	20.92	91.42	15.68	91.38	15.97
Letter-high	92.40	100.00	**90.27**	15.43	**90.27**	15.67	89.07	16.43	89.24	18.95	88.89	30.92	87.38	27.53	85.51	22.32
GREC	99.55	100.00	**99.36**	9.20	**99.36**	9.24	96.91	12.99	97.18	13.55	98.55	24.36	99.00	22.62	92.18	23.07
Fingerprint	82.29	100.00	**82.43**	10.99	81.93	12.10	72.46	10.46	81.18	12.86	81.18	25.65	81.89	22.31	78.25	19.00
AIDS	99.25	100.00	99.45	1.87	**99.50**	1.90	99.10	1.42	99.45	1.60	99.40	3.87	99.10	11.34	99.10	2.24
Protein	69.17	100.00	55.33	31.78	**59.00**	35.57	47.17	41.11	52.50	49.48	47.17	49.22	46.33	38.48	39.17	44.11
Mutagenicity	72.49	100.00	70.99	25.04	**71.94**	25.31	69.56	27.93	69.82	31.18	69.68	41.63	68.80	33.44	65.02	31.22
Coil-rag	93.62	100.00	89.41	22.89	**89.79**	27.84	85.79	27.06	87.69	29.17	87.87	40.71	87.18	35.27	81.97	34.81
Coil-del	89.23	100.00	77.97	38.46	**78.10**	38.51	65.44	40.70	69.95	48.07	70.31	55.58	75.26	43.66	65.03	47.72
Promedio	89.26	100.00	85.83	16.84	**86.25**	17.90	81.57	19.28	83.79	22.19	83.43	30.14	83.54	25.81	79.62	24.91
5-NN																
Letter-low	99.42	100.00	99.33	3.31	**99.51**	3.38	98.62	4.95	98.71	5.28	99.24	8.66	98.53	7.81	97.82	8.58
Letter-med	95.16	100.00	**94.18**	11.45	93.11	11.51	92.40	13.00	92.31	14.49	92.58	20.33	91.82	15.68	90.67	15.97
Letter-high	93.29	100.00	**91.38**	17.44	91.16	18.17	88.67	19.79	89.16	21.72	88.98	30.85	88.76	27.53	86.27	22.32
GREC	99.55	100.00	**99.09**	12.56	**99.09**	12.65	96.36	16.72	97.09	17.27	95.73	24.57	96.09	22.62	91.09	23.07
Fingerprint	83.00	100.00	82.21	12.14	81.89	13.59	80.14	10.96	82.14	13.13	82.11	24.90	**83.29**	22.31	78.68	19.00
AIDS	99.05	100.00	99.50	2.08	**99.55**	2.12	99.25	1.82	99.30	1.94	99.50	3.90	99.05	11.34	98.30	2.24
Protein	62.83	100.00	52.67	34.22	**58.33**	37.94	40.67	46.39	45.33	54.50	45.00	48.43	43.33	38.48	33.33	44.11
Mutagenicity	73.76	100.00	72.68	29.26	**72.86**	31.59	69.84	30.83	69.89	33.46	68.39	40.26	69.15	33.44	65.69	31.22
Coil-rag	91.69	100.00	88.18	27.80	**88.38**	28.83	83.77	32.11	85.92	34.33	84.38	41.21	82.46	35.27	76.18	34.81
Coil-del	85.10	100.00	75.54	44.48	**75.69**	44.54	63.67	46.09	66.72	52.59	65.97	55.52	68.33	43.66	61.21	47.72
Promedio	88.29	100.00	85.48	19.47	**85.96**	20.43	81.34	22.27	82.66	24.87	82.19	29.86	82.08	25.81	77.92	24.91

Con la finalidad de validar los resultados de clasificación en esta comparación experimental, la prueba de *Friedman* y el *post-hoc Bergmman-Hommel* fueron aplicados sobre estos resultados con un nivel de significancia $\alpha = 0.05$. Los diagramas CD obtenidos de estas pruebas estadísticas se muestran en las figuras 5.14 y 5.15. De manera análoga a secciones anteriores, estas pruebas fueron divididas en dos partes. En la primera parte se evaluaron los algoritmos de tipo *wrapper*; mientras que en la segunda parte se evaluaron los algoritmos de tipo *filter*, además de la mejor solución obtenida en la primera parte.

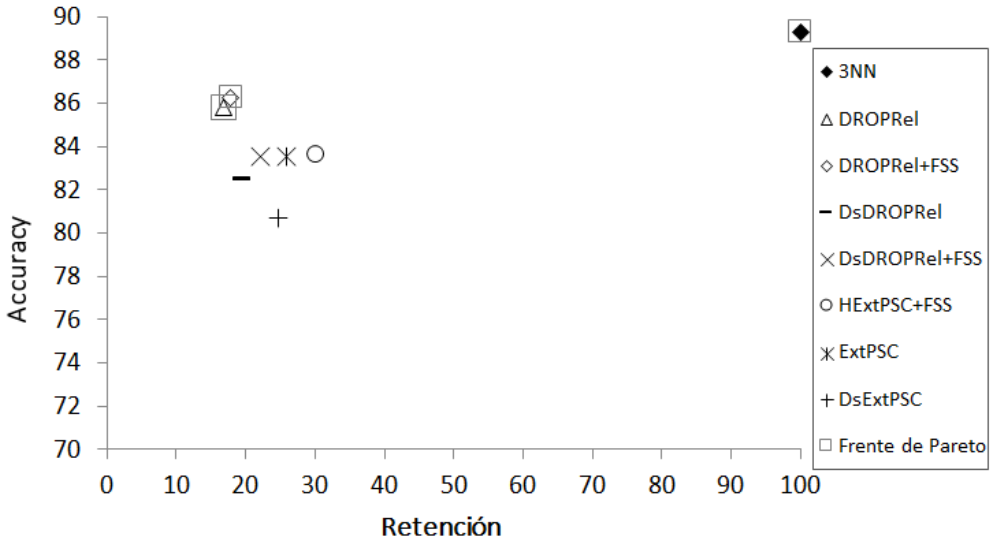

Figura 5.12: Comparación de las soluciones propuestas en los diferentes enfoques de selección, usando el clasificador 3-NN.

Con base en los diagramas CD de la figura 5.14, observamos que el mejor algoritmo en el ranking es DROPRel+FSS, seguido por DROPRel. El *post-hoc Bergmman-Hommel* indica que ambos algoritmos obtienen una mejora significativa con respecto a DsDROPRel y DsDROPRel+FSS (soluciones del espacio n-dimensional) para los clasificadores 1 y 3-NN; mientras que comparados con HExtPSC+FSS la solución híbrida, DROPRel y DROPRel+FSS son estadísticamente mejores en 3 y 5-NN. Por último, es posible observar que las soluciones propuestas en el espacio n-dimensional e híbrida obtienen resultados de clasificación estadísticamente similares.

En la figura 5.15, se muestran los diagramas CD para los algoritmos *filter* y

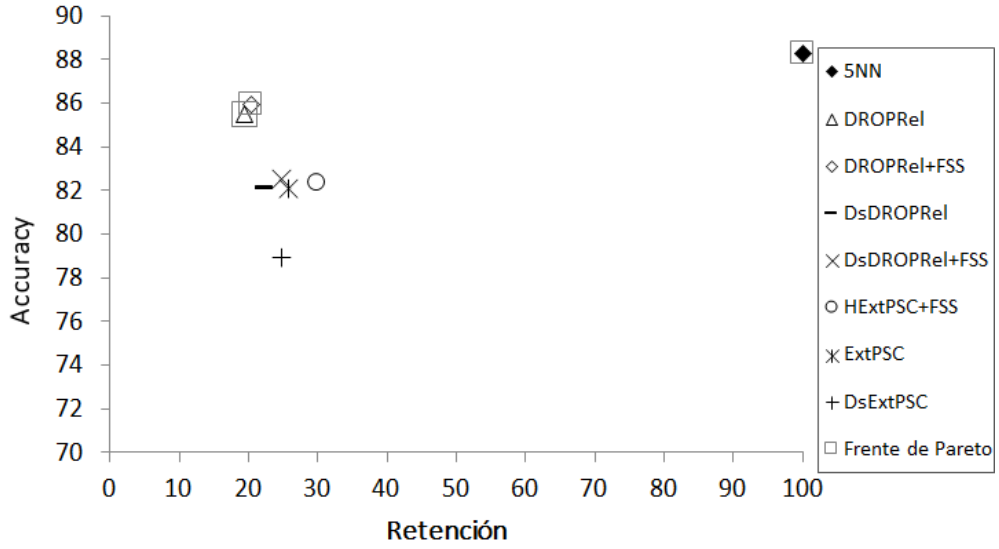

Figura 5.13: Comparación de las soluciones propuestas en los diferentes enfoques de selección, usando el clasificador 5-NN.

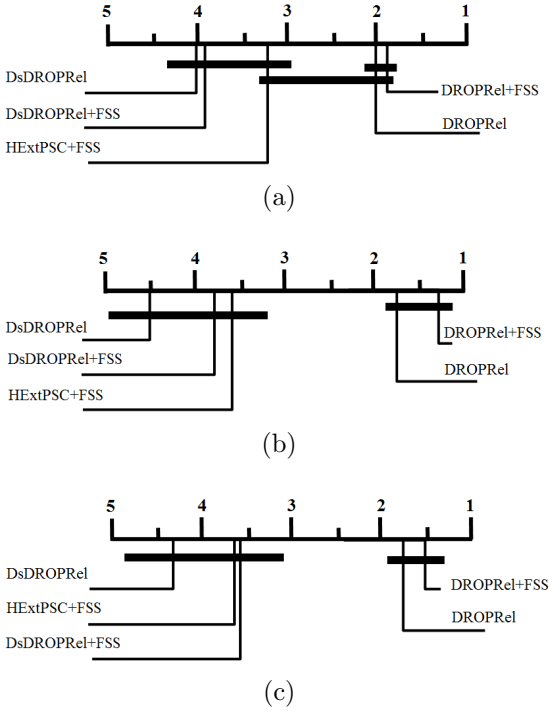

Figura 5.14: Diagramas CD para las soluciones propuestas en los distintos enfoques de selección de instancias en el espacio de los grafos: (a) 1-NN, b) 3-NN y (c) 5-NN.

DROPRel+FSS el mejor en el primer análisis estadístico. Con base en estos diagra-

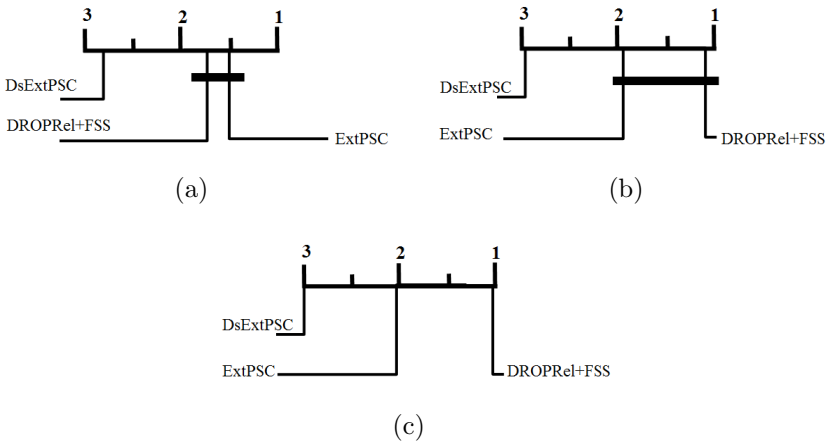

Figura 5.15: Diagramas CD para los algoritmos filter y DROPRel+FSS: (a) 1-NN, b) 3-NN y (c) 5-NN.

mas, el algoritmo ExtPSC es el mejor ubicado en el ranking para 1-NN, no obstante en 3 y 5-NN el mejor es DROPRel+FSS. El *post-hoc Bergmman-Hommel* indica que ExtPSC y DROPRel+FSS son estadísticamente similares en 1 y 3-NN, mientras que en 5-NN existe diferencia significativa entre ambos algoritmos. Finalmente, se observa que ExtPSC y DROPRel+FSS son estadísticamente mejores que DsExtPSC en todos los clasificadores, este último algoritmo es el mejor de tipo $filter$ presentado en el espacio de n-dimensional.

En general, el algoritmo con mejor desempeño en calidad de clasificación es el algoritmo *wrapper* DROPRel+FSS, pero requiere tiempos de ejecución mayores. Por otro lado, se puede optar por los algoritmos DROPRel o ExtPSC, ambos con un tiempo de ejecución mucho menor. El primero de estos, de tipo *wrapper*, con resultados de clasificación estadísticamente similares a los de DROPRel+FSS en todos los clasificadores utilizados, mientras que el segundo, de tipo *filter*, con resultados estadísticamente similares a DROPRel+FSS en los clasificadores 1 y 3-NN.

5.6. Discusión

En esta tesis se exploraron diferentes enfoques para solucionar el problema de la selección de instancias en bases de datos de grafos. Primero, se exploró el espacio de representación de los grafos, en el cual no se han reportado en la literatura

algoritmos que solucionen este problema, y los algoritmos más cercanos como los de *selección de prototipos para embedding de grafos*, no son útiles para la selección de instancias. Adicionalmente, en nuestra evaluación experimental se mostró que las soluciones propuestas en este espacio de representación, obtienen mejores resultados de clasificación que algoritmos de selección de instancias adaptados por nosotros para trabajar en el espacio de los grafos.

Posteriormente, nos dimos a la tarea de explorar el espacio n-dimensional, donde una gran cantidad de algoritmos de selección de instancias se han desarrollado. Estos algoritmos surgen como una alternativa al problema abordado en esta tesis, dado que es posible generar un espacio de representación n-dimensional usando el *embedding de grafos vía disimilaridad*. En este nuevo espacio de representación, nuestras soluciones propuestas obtienen mejores resultados de clasificación que algunos de los algoritmos más exitosos reportados en la literatura.

Un tercer enfoque fue explorado, el cual es una combinación de los dos primeros. Los resultados de clasificación obtenidos para éste son similares a los alcanzados por nuestras mejores soluciones en el espacio n-dimensional. La principal razón de estos resultados, es que el conjunto reducido, obtenido por la selección de instancias, es usado para la clasificación en el espacio n-dimensional.

Finalmente, comparamos los algoritmos propuestos en estos enfoques con el propósito de encontrar la mejor solución al problema planteado en esta tesis. De acuerdo a lo mostrado experimentalmente, los algoritmos de selección de instancias en el espacio de los grafos obtienen los mejores resultados de clasificación, superando a las soluciones propuestas en el espacio n-dimensional e híbridas.

Capítulo 6

Conclusiones

La selección de instancias es un problema de gran importancia en aquellos dominios donde una buena calidad de clasificación y un tiempo de respuesta reducido son requeridos. No obstante, para bases de datos de grafos, en la literatura no se reportan algoritmos que solucionen este problema, pues los algoritmos de selección de instancias existentes fueron desarrollados para espacios n-dimensionales.

En esta tesis se propusieron y compararon diferentes soluciones para la selección de instancias en bases de datos grafos usando tres enfoques. En el primer enfoque, basado en una selección en el espacio de los grafos, se presentaron los algoritmos de tipo *wrapper* DROPRel y DROPRel+FSS, además del algoritmo de tipo *filter* ExtPSC. El primero de los tres algoritmos presentados, selecciona los grafos menos disimilares a los de su clase, mientras el segundo, en una segunda etapa, incluye grafos descartados por DROPRel que mejoren la calidad del conjunto de entrenamiento; el tercer algoritmo por otro lado, selecciona los grafos ubicados en la frontera de decisión y algunos al interior de la clase. En el segundo enfoque, los tres algoritmos propuestos en el espacio de los grafos fueron adaptados para trabajar en el espacio n-dimensional generado por el *embedding de grafos vía disimilaridad*. Finalmente, el tercer enfoque, denominado de selección híbrida, fue diseñado a partir de una combinación de los dos primeros. En este enfoque, el algoritmo DROPRel (o ExtPSC) es aplicado en el espacio de los grafos y una inclusión de instancias es realizada en el espacio n-dimensional.

Con base en los resultados experimentales reportados en este trabajo de investigación, los algoritmos propuestos en el espacio de los grafos permiten obtener una

selección de (grafos) instancias con los mejores resultados de clasificación comparados con los algoritmos del espacio n-dimensional e híbridos. Siendo el algoritmo DROPRel+FSS la mejor solución propuesta en cuanto a calidad de clasificación. Dado que este algoritmo es costoso en tiempo, se puede optar por soluciones como DROPRel o ExtPSC, si lo que se busca es mayor velocidad de ejecución; ambas soluciones producen resultados estadísticamente similares a DROPRel+FSS.

Con los algoritmos de selección de instancias propuestos en el espacio de los grafos, se presentan varias alternativas para reducir el conjunto de entrenamiento sin afectar demasiado la calidad de clasificación, cumpliendo así con el objetivo planteado en esta tesis.

6.1. Contribuciones

Las aportaciones de este trabajo de investigación son tres algoritmos de selección de instancias en el espacio de los grafos, además de los algoritmos propuestos en el espacio n-dimensional e híbridos:

1. El algoritmo DROPRel de tipo *wrapper*, el cual descarta los grafos del conjunto de entrenamiento que no afectan la calidad de clasificación, analizando primero los grafos más disimilares en promedio a los grafos de su clase para finalizar con los grafos menos disimilares a los de su clase.

2. El algoritmo DROPRel+FSS, de tipo *wrapper*, está basado en una etapa de exclusión seguida por una de inclusión. En la primera etapa se utiliza el algoritmo DROPRel para reducir el conjunto entrenamiento. Posteriormente, en la segunda etapa se incluyen, mediante el algoritmo FSS, aquellos grafos descartados por DROPRel que mejoren la calidad del conjunto de entrenamiento.

3. El algoritmo ExtPSC, de tipo *filter*, está basado en agrupamiento para seleccionar grafos ubicados cerca de la frontera de decisión y algunos al interior de la clase. En la selección de los grafos frontera se seleccionan aquellos que son más disimilares a los grafos de otras clases y más similares a los grafos de su misma clase.

4. Los algoritmos DsDROPRel, DsDROPRel+FSS y DsExtPSC propuestos en el espacio n-dimensional, que son adaptaciones de los algoritmos propuestos en el espacio de los grafos.

5. Los algoritmos híbridos HDroprel+FSS e HExtPSC+FSS, desarrollados a partir de combinaciones de los dos primeros enfoques de selección de instancias.

6.2. Trabajo futuro

Los resultados obtenidos en esta tesis nos motivan a realizar como trabajo futuro lo siguiente:

1. Los algoritmos de selección híbridos presentados fueron explorados en una sola dirección, la cual consiste en una selección primero en el espacio de los grafos y luego en el espacio n-dimensional; obteniendo resultados alentadores. Con base en esto y dado que una selección de instancias en el espacio de los grafos permite obtener los mejores resultados de clasificación, es de nuestro interés explorar una selección híbrida que considere una selección en el espacio de los grafos posterior a una selección en el espacio n-dimensional generado por el *embedding de grafos*, enfocándonos principalmente en aquellas estrategias de selección de tipo $filter$.

2. La medida de disimilaridad utilizada en nuestros algoritmos está basada en una solución subóptima de la distancia de edición de grafos (GED por sus siglas en inglés *Graph Edit Distance*). Aunque esta solución obtiene una buena aproximación de la GED en un tiempo de ejecución menor que diversas soluciones reportadas en la literatura, es de nuestro interés probar otras soluciones subóptimas que permitan reducir aún más el costo computacional para el cálculo de la GED; manteniendo una precisión similar de los valores obtenidos por la medida utilizada en este trabajo de investigación. Al tener una solución de la GED con un costo mucho menor, permitirá que los tiempos totales de la selección de instancias se reduzcan.

6.3. Publicaciones

Derivado de este trabajo de investigación se publicó el siguiente artículo:

1. Magdiel Jiménez-Guarneros, Jesús Ariel Carrasco-Ochoa, José Fco. Martínez Trinidad. **Prototype Selection for Graph Embedding using Instance Selection**. Mexican Conference on Pattern Recognition 2015, LNCS 9116, pp. 81-90, 2015. Springer-Verlag.

Bibliografía

Auwatanamongkol, S. (2007). Inexact graph matching using a genetic algorithm for image recognition. *Pattern Recognition Letters*, 28(12):1428 – 1437.

Berretti, S., Del Bimbo, A., and Vicario, E. (2001). Efficient matching and indexing of graph models in content-based retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(10):1089–1105.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56.

Borgwardt, K. M., Petri, T., Vishwanathan, S. V. N., and Kriegel, H. (2007). An efficient sampling scheme for comparison of large graphs. In Frasconi, P., Kersting, K., and Tsuda, K., editors, *Mining and Learning with Graphs, MLG 2007, Firence, Italy, August 1-3, 2007, Proceedings*.

Borzeshi, E. Z., Piccardi, M., Riesen, K., and Bunke, H. (2013). Discriminative prototype selection methods for graph embedding. *Pattern Recognition*, 46(6):1648 – 1657.

Brighton, H. and Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.*, 6(2):153–172.

Bunke, H. (2000). Recent developments in graph matching. In *ICPR*, pages 2117–2124. IEEE Computer Society.

Bunke, H. (2006). *Graph-theoretic Approach to Enterprise Network Dynamics*. Springer, Berlin.

Bunke, H., Foggia, P., Guidobaldi, C., Sansone, C., and Vento, M. (2002). A comparison of algorithms for maximum common subgraph on randomly connected graphs. In *Lecture Notes on Computer Science*, volume 2396, pages 123–132. Springer-Verlag, Heidelberg.

Bunke, H. and Riesen, K. (2008). Graph classification based on dissimilarity space embedding. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342, pages 996–1007. Springer Berlin Heidelberg.

Bunke, H. and Riesen, K. (2011). Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recogn.*, 44(5):1057–1067.

Chierichetti, F., Kumar, R., Lattanzi, S., Mitzenmacher, M., Panconesi, A., and Raghavan, P. (2009). On compressing social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 219–228.

Chou, C., Kuo, B., and Chang, F. (2006). The generalized condensed nearest neighbor rule as A data reduction method. In *18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China*, pages 556–559.

Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of Graph Matching in Pattern Recognition.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27.

Cross, A. D., Wilson, R. C., and Hancock, E. R. (1997). Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953 – 970.

de Assis Zampirolli, F., Stransky, B., Lorena, A., and de Melo Paulon, F. (2010). Segmentation and classification of histological images - application of graph analysis and machine learning methods. In *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*, pages 331–338.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Fankhauser, S., Riesen, K., and Bunke, H. (2011). Speeding up graph edit distance computation through fast bipartite matching. In *Graph-Based Representations in Pattern Recognition*, volume 6658, pages 102–111. Springer Berlin Heidelberg.

Fernández, M.-L. and Valiente, G. (2001). A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6?7):753 – 758.

Fischer, A., Suen, C. Y., Frinken, V., Riesen, K., and Bunke, H. (2015). Approximation of graph edit distance based on hausdorff matching. *Pattern Recognition*, 48(2):331 – 343.

Gao, X., Xiao, B., Tao, D., and Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and Applications*, 13(1):113–129.

García, S., Luengo, J., and Herrera, F. (2015). Instance selection. In *Data Preprocessing in Data Mining*, volume 72 of *Intelligent Systems Reference Library*, pages 195–243. Springer International Publishing.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gregory, L. and Kittler, J. (2002). Using graph search techniques for contextual colour retrieval. In Caelli, T., Amin, A., Duin, R., de Ridder, D., and Kamel, M., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin Heidelberg.

Gärtner, T., Flach, P., and Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *IN: CONFERENCE ON LEARNING THEORY*, pages 129–143.

Han, P. Y., San, H. F., and Yin, O. S. (2012). Face recognition using a kernelization of graph embedding. 6(2):460 – 464.

Harchaoui, Z. and Bach, F. (2007). Image classification with segmentation graph kernels. IEEE Computer Society.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516.

Hernandez-Leal, P., Carrasco-Ochoa, J. A., Martínez-Trinidad, J., and Olvera-Lopez, J. A. (2013). Instancerank based on borders for instance selection. *Pattern Recognition*, 46(1):365 – 375.

Jiang, X., Munger, A., and Bunke, H. (2001). An median graphs: properties, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(10):1144–1151.

Justice, D. and Hero, A. (2006). A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1200–1214.

Livi, L., Rizzi, A., and Sadeghian, A. (2014). Optimized dissimilarity space embedding for labeled graphs. *Information Sciences*, 266(0):47 – 64.

Lozano, M. and Escolano, F. (2006). Protein classification by matching and clustering surface graphs. *Pattern Recognition*, 39(4):539 – 551.

Marchiori, E. (2010). Class conditional nearest neighbor for large margin instance selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):364–370.

Marcialis, G., Roli, F., and Serrau, A. (2007). Graph-based and structural methods for fingerprint classification. In *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52 of *Studies in Computational Intelligence*, pages 205–226.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Nikolaidis, K., Goulermas, J., and Wu, Q. (2011). A class boundary preserving algorithm for data condensation. *Pattern Recognition*, 44(3):704 – 715.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., and Kittler, J. (2010). A review of instance selection methods. *Artif. Intell. Rev.*, 34(2):133–143.

Olvera-López, J., Carrasco-Ochoa, J., and Martínez-Trinidad, J. (2008). Prototype selection via prototype relevance. In *Progress in Pattern Recognition, Image Analysis and Applications*, volume 5197 of *Lecture Notes in Computer Science*, pages 153–160. Springer Berlin Heidelberg.

Olvera-López, J., Carrasco-Ochoa, J., and Martínez-Trinidad, J. (2010). A new fast prototype selection method based on clustering. *Pattern Analysis and Applications*, 13(2):131–141.

Olvera-López, J. A., Trinidad, J. F. M., Carrasco-Ochoa, J. A., and Kittler, J. (2009). Prototype selection based on sequential search. *Intell. Data Anal.*, 13(4):599–631.

Pekalska, E., Duin, R. P. W., and Paclík, P. (2006). Prototype selection for dissimilarity-based classifiers. *Pattern Recogn.*, 39(2):189–208.

Raicharoen, T. and Lursinsap, C. (2005). A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm. *Pattern Recognition Letters*, 26(10):1554 – 1567.

Ramon, J. and Gärtner, T. (2003). Expressivity versus efficiency of graph kernels. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74.

Riesen, K. (2009). Classification and clustering of vector space embedded graphs. phd thesis, department of computer science and applied mathematics-university bern.

Riesen, K. and Bunke, H. (2008). Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342, pages 287–297. Springer Berlin Heidelberg.

Riesen, K. and Bunke, H. (2009a). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950 – 959.

Riesen, K. and Bunke, H. (2009b). Dissimilarity based vector space embedding of graphs using prototype reduction schemes. In Perner, P., editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 5632 of *Lecture Notes in Computer Science*, pages 617–631. Springer Berlin Heidelberg.

Riesen, K. and Bunke, H. (2009c). Graph classification based on vector space embedding. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(06):1053–1081.

Riesen, K. and Bunke, H. (2015). Improving bipartite graph edit distance approximation using various search strategies. *Pattern Recognition*, 48(4):1349 – 1363.

Riesen, K., Neuhaus, M., and Bunke, H. (2007). Graph embedding in vector spaces by means of prototype selection. In *Graph-Based Representations in Pattern Recognition*, volume 4538, pages 383–393. Springer Berlin Heidelberg.

Serratosa, F. (2014). Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45(0):244 – 250.

Sharma, H., Alekseychuk, A., Leskovsky, P., Hellwich, O., Anand, R., Zerbe, N., and Hufnagl, P. (2012). Determining similarity in histological images using graph-theoretic description and matching methods for content-based image retrieval in medical diagnostics. *Diagnostic Pathology*.

Suganthan, P. (2002). Structural pattern recognition using genetic algorithms. *Pattern Recognition*, 35(9):1883 – 1893.

Vallejo, C. G., Troyano, J. A., and Ortega, F. J. (2010). Instancerank: Bringing order to datasets. *Pattern Recogn. Lett.*, 31(2):133–142.

Wallis, W., Shoubridge, P., Kraetz, M., and Ray, D. (2001). Graph distances using graph union. *Pattern Recognition Letters*, 22(6?7):701 – 704.

Wilson, D. and Martinez, T. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, 2(3):408–421.

Zhao, K.-P., Zhou, S.-G., Guan, J.-H., and Zhou, A.-Y. (2003). C-pruner: an improved instance pruning algorithm. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 1, pages 94–99 Vol.1.