# Some results about generalisation of graphs embedded in metric spaces

V. Estruch C. Ferri J. Hernández-Orallo M.J. Ramírez-Quintana

DSIC, Univ. Politècnica de València , Camí de Vera s/n, 46020 València, Spain.
{vestruch,cferri,jorallo,mramirez}@dsic.upv.es

**Abstract.** Distances and similarity functions between structured datatypes, such as graphs, have been widely employed in machine learning since they are able to identify similar cases or prototypes from which decisions can be made. In these distance-based methods the justification of the labelling of a new case $a$ is usually based on expressions such as "label($a$)=label($b$) because case $a$ is similar to case $b$". However, a more general or meaningful pattern, such as "because case $a$ has properties $x$ and $y$ (as $b$ has)" is usually more difficult to find. Furthermore, the connection of this pattern with the original distance-based method might be unclear, or even inconsistent. The relationship between the concept of distance (or similarity), generalisation and pattern languages has been studied in a general way in [5]. In this paper we analyse the particular case of graphs embedded in metric spaces. We also study how to characterise consistent generalisation operators for one given metric space of graphs. Then, we analyse some properties related to the pattern language and the metric space employed.

**Keywords**: distance-based methods, generalisation operators, graph-based representation, graph-edit distance, metric spaces.

## 1 Introduction

While in some learning problems the data can be described by a single-fixed row of nomimal or numerical attributes, in most others, especially from new challenging scenarios such as biomedicine and web mining, a more expressive representation language is needed. In fact, in many cases, the data can be directly represented as a graph. For instance, in molecule classification, the labelled vertices of the graph would correspond to atoms and the edges to chemical bonds.

In general, for structured-data domains (e.g. graph-based instances), those properties inherent to the sort of data (e.g. common subgraphs, trees, cycles, etc.), might be important to achieve a competitive solution. For this reason, the approaches based on flattening the structured data in order to obtain a propositional representation might not be suitable for this kind of problems since propositionalisation implies the loss of the original data structure. In addition, other drawbacks of these techniques is that flattening data could not be a trivial

task, specially when data is highly-structured as it happens in web mining or in molecule classification.

This circumstance has motivated that some learning techniques which directly deal with structured data have been developed (multi-relational data mining [3]). Among them, we focus on distance-based methods, which unlike the ILP approaches, do not give an explanation of their predictions.

Roughly speaking, it is due to the matches between two objects (e.g. two molecules) are encoded by a number (their distance). Unfortunately, model comprehensibility would be useful in some contexts. Imagine, for instance, in molecule classification, how interesting it would be to describe a cluster of molecules by saying what chemical structures these molecules have in common instead of saying that they are closed according to a certain distance measure used in the clustering process. Providing the possibility of this kind of descriptions for a distance-based algorithm would imply to incorporate a pattern language and some generalisation operators such that the patterns would be seen as comprehensible explanations of the generalisations. But in this case, an immediate issues arises: the model (expressed in a hopeful comprehensible pattern language) which generalises a set of elements could not be consistent with the distance employed.

Initially, the issue above was considered in [4] by introducing the notion of distance-based binary generalisations. In [5], a more general framework in order to handle $n$-ary generalisation operators and to introduce the notion of minimal distance-based generalisations is presented.

In this paper, we use some of the concepts introduced in [5], to study generalisation operators and pattern languages for graph-based representations embedded in metric spaces. For this purpose, we consider two graph distance functions: the first one is described in [2], and then we propose a second distance which is a bit more general. Next, a pattern language for graphs is introduced. This language is utilised to represent, in a comprehensible way, the generalisation computed by the generalisation operator. Next, for each metric space, a special kind of generalisation operator is defined according to our framework. From these generalisation operators, we will show that some interesting aspects such as the complexity of the metric space and the expressivity of the pattern language will be deduced.

The paper is organised as follows. Section 2 presents both the notation and the concepts of graphs used through this paper. Our generalisation setting is introduced in Section 3 and it is applied in Section 4, when data is represented by means of graphs. Finally, we finish with some conclusions and future lines of work in Section 5.

## 2 Preliminaries

In this section we briefly review some basic concepts about graphs and graph distances. For any concept not explicitly included we refer the reader to [2].

A graph is a 4-tuple $G = (V, E, \mu, \nu)$ where $V$ is a finite set of vertices, $E$ is a set of edges (each one denoted as a pair of vertices belonging to $V \times V$), and $\mu$ and $\nu$ are functions which assign labels to vertices and edges respectively. The number of nodes of a graph $G = (V, E, \mu, \nu)$ is given by $|V|$ and it is denoted as $|V_G|$. The number of edges of a graph $G$ is given by $|E|$ and is denoted as $|E_G|$. Given a graph $G = (V, R, \mu, \nu)$, a subgraph of $G$ is a graph $S = (V_S, E_S, \mu_S, \nu_S)$ such that $V_S \subseteq V$, $E_S \subseteq E \cap (V_S \times V_S)$, and $\mu_S$ and $\nu_S$ are the restrictions of $\mu$ and $\nu$ to $V_S$ and $E_S$ respectively, that is $\mu_S(v) = \mu(v)$ (res. $\nu_S(e) = \nu(e)$) if $v \in V_S$ (res. $e \in E_S$) and undefined in otherwise.

Two graphs $G_1 = (V_1, E_1, \mu_1, \nu_1)$ and $G_2 = (V_2, E_2, \mu_2, \nu_2)$ are isomorphic if there is a bijection $\phi : V_1 \rightarrow V_2$ such that for every pair of vertices $v_i, v_j \in V_1, (v_i, v_j) \in E_1$, if and only if $(\phi(v_i), \phi(v_j)) \in E_2$ and $\phi(v_i), \phi(v_j) \in V_2$. Let $G$, $G_1$ and $G_2$ be graphs. $G$ is a common subgraph of $G_1$ and $G_2$ if it is a subgraph of $G_1$ and $G_2$. A common subgraph $G$ of $G_1$ and $G_2$ is maximal, denoted as $mcs(G_1, G_2)$, if there exists no other common subgraph $G'$ that has more nodes than $G$. A subgraph $G_1$ of a graph $G = (V, E, \mu, \nu)$ is said to be induced by a set of vertices $W \subseteq V$ if for any pair of vertices $w_1$ and $w_2$ of $W$, $(w_1, w_2)$ is an edge of $G_1$ if and only if $(w_1, w_2) \in E$, that is, $G_1$ is isomorphic to $G$. The concepts of common subgraph and maximal common subgraph are trivially extended to subgraphs induced by a set of vertices. We will call $vimcs(G_1, G_2)$ to the maximal common subgraph of $G_1$ and $G_2$ induced by a set of vertices. Figure 1 illustrates with an example the above concepts. Note that, in general, there can be more than one maximal common subgraph induced by a set of vertices.

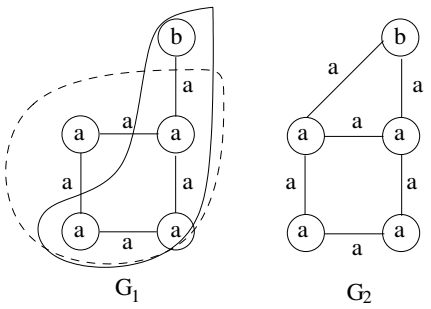


**Fig. 1.** $G_1$ **is the maximal common subgraph of** $G_1$ **and** $G_2$ **whereas the subgraphs marked with dashed points and lines are the two maximal common subgraphs of** $G_1$ **and** $G_2$ **induced by a set of vertices.**

In [2] a graph distance based on the maximal common subgraph is introduced. We call this distance $d_1$ and it is defined in terms of the minimal cost mapping

transforming one graph into other. It is shown in [2], that the distance $d_1$ of two graphs $G_1$ and $G_2$ can be expressed as:

$$d_1(G_1, G_2) = |V_{G_1}| + |V_{G_2}| - 2|V_{vimcs(G_1,G_2)}|$$

By modifying the definition of cost mapping employed in [2] we have derived a more general distance $d_2$ which can be calculated as:

$$d_2(G_1, G_2) = |V_{G_1}| + |V_{G_2}| - 2|V_{mcs(G_1,G_2)}| + |E_{G_1}| + |E_{G_2}| - 2|E_{mcs(G_1,G_2)}|$$

Regarding the graphs $G_1$ and $G_2$ in Figure 1, we have $|V_{G_1}| = |V_{G_2}| = 5$, $|E_{G_1}| = 5$, $|E_{G_2}| = 6$, $V_{mcs(G_1,G_2)} = 5$, $E_{mcs(G_1,G_2)} = 5$ and $V_{vimcs(G_1,G_2)} = 4$. Hence, $d_1(G_1, G_2) = 2$ and $d_2(G_1, G_2) = 1$.

Note that the computation of both distances is an $NP$ problem because the computation of *mcs* is needed. However, we are not interested in its calculus but we will use them to illustrate that extracting meaningful patterns when the distance $d_2$ is used, is easier than using the distance $d_1$.

## 3 Distance-based generalisation operators

In this section we present the main concepts related to our proposal of a generalisation framework based on distances. For a more detailed presentation see [5].

Our approach aims to define generalisation operators for data embedded in a metric space $(X, d)$. These operators are denoted as $\Delta(E)$, where $E$ is a finite set of elements ($|E| > 2$) of $X$ to be generalised. In principle, a generalisation of $E$ will be a particular set in $2^X$ containing $E$. But, for the sake of comprehensibility, the generalisation computed by $\Delta(E)$ will be expressed by a pattern $p$ belonging to a pattern language $\mathcal{L}$. In fact, every pattern $p$ represents a set of elements of $X$ and it is denoted by $Set(p)$. In this way, we can say that an element $x \in X$ is covered by a pattern $p$, if $x \in Set(p)$.

Additionally, if for every $E$, $\Delta(E)$ computes a generalisation of $E$ "explaining" the distances among the elements of $E$, we will say that $\Delta$ is a distance-based generalisation operator. Then, the objective will be to find possible distance-based generalisation operators for the metric space $(X, d)$.

For this purpose, we will focus on a particular kind of metric spaces, the connected ones. Informally speaking, this property means that given two elements in the metric space, we can always go from one to the other one through elements belonging to the space such that these elements are at a minimal distance. In order to formally define what a connected space is, we need to introduce the notion of $\delta$-path and the length of a $\delta$-path.

Given a metric space $(X, d)$, $I(X) = inf\{d(x, y) : \forall x, y \in X, x \neq y\}$ denotes the infimum distance of $X$. Let $\delta$ be a real positive number such that $\delta \geq I(X)$. Then,

- if $I(X) > 0$, a $\delta - path$ is a sequence of elements $P = \{x_i\}_{i=0}^{n>0}, \forall x_i \in X$, if $d(x_i, x_{i+1}) \leq \delta$ for all $0 \leq i \leq n - 1$.

– if $I(X) = 0$, a $\delta - path$ is a sequence of elements $P$ belonging to $X$ if $P$ is the image of a continuous function $\gamma$ defined over the closed interval $[0, 1]$ if the space is continuous or over the rational number in $[0, 1]$ otherwise, such that $\gamma([0, 1]) = P$.

If $P = \{x_i\}_{i=0}^{n>0}, x_i \in X$ is a $\delta - path$ then the length of $P$, denoted as $L(P)$, is defined as follows:

$$L(P) = \begin{cases} \sum_{i=0}^{n-1} d(x_i, x_{i+1}), \text{ if } \delta > 0 \\ sup\{\sum_{i=0}^{n-1} d(\gamma(t_i), \gamma(t_{i+1})) : \forall n \in N, 0 = t_0 < \cdots < t_n = 1\}, \text{ if } \delta = 0 \end{cases}$$

Let $(X, d)$ be a metric space, two elements $x, y \in X$ are connected by a $\delta$-path or equivalently, are $\delta$-path connected, if there exists a $\delta$-path $P = \{x_i\}_{i=0}^{n>0}$ such that $x_0 = x$ and $x_n = y$. Next, we will say that $X$ is a connected metric space, if for every pair of elements $x$ and $y$ belonging to $X$, they are $\delta$-path connected with $\delta \geq I(X)$. From this definition, we can say that a set $S \subseteq X$ is connected, if for every pair of elements in $S$ they are $\delta$-path connected, being $\delta = I(X)$. In what follows, we will use the term of $x$-connected space meaning that the metric space is connected and its $I(\cdot) = x$.

The notion of connected spaces and sets plays a fundamental role in our approach since it permits us to reject much too specific generalisations (see Example 1).

*Example 1.* Let us suppose we are clustering graphs with the distance $d_1$ defined in Section 2. Imagine that each graph represents an organic compound and we would be interested in extracting some patterns saying which kind of molecules can be found in a cluster. One of the obtained clusters consisting of two molecules ($m_1$ and $m_2$) is depicted on the top Figure 2.

Let us obtain a pattern explaining the data distribution in this cluster. For this purpose, one could think on the pattern $p$ (bottom-right Figure 2) saying all the molecules with a cyclopropane structure and an extra atom. But, is this pattern much too specific? Considering that the molecules are really graphs in the space $(G, d_1)$, we could consider that the pattern $p$ overfits the data since the cyclopropane molecule, which would be placed in the"middle"[1] of $m_1$ and $m_2$, that is, $d_1(m_1, cyclopropane) = d_1(cyclopropane, m_2) = 1$, is not covered by this pattern. Perhaps, a more natural pattern would be that one saying "all the molecules built from cyclopropane".

The last reasoning can be modelled in terms of connections. We know that $(G, d_1)$ is a 1-connected space (see Proposition 1 in Appendix), and clearly, the set given by $Set(p)$ is not connected because the elements $m_1$ and $m_2$ are not connected by means of a 1-path included in $Set(p)$. That is, $m_1$ and $m_2$ are, at least, 2-path connected since $d_1(m_1, m_2) = 2$. However, $Set($"all the molecules built from cyclopropane"$)$ would be connected.

Now, we are in conditions of introducing the formal definition of a generalisation operator.

[1] Formally, given three elements $x$, $y$ and $z$ belonging to a metric space $(X, d)$, we say that $z$ is in the "middle" of $x$ and $y$ if $d(x, y) = d(x, z) + d(z, y)$.

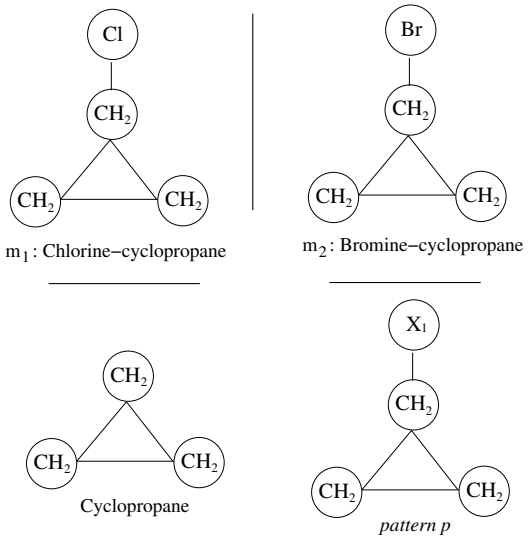


**Fig. 2.** The pattern $p$ does not cover the cyclopropane molecule.

**Definition 1.** ***(Generalisation operator)*** *Let $(X, d)$ be a connected metric space and let $\mathcal{L}$ be a pattern language. For every finite set $E$ of elements in $X$, a generalisation operator $\Delta$ is a function $\Delta : E \to p \in \mathcal{L}$ such that $E \subset Set(p)$ and $Set(p)$ is a connected set.*

Note that this latter definition permits to avoid some specific generalisations but does not ensure that the generalisation computed by $\Delta$ informs about the distances among the elements in $E$. In other words, given two elements $x, y \in X$, we could find a connected set covering $x$ and $y$ but excluding those elements $z \in X$ placed in the middle of $x$ and $y$.

In order to define distance-based generalisations, the concept of "nerve" of a set of elements $E$ is needed. In this way, a nerve of $E$, denoted by $N(E)$, is simply a connected[2] graph whose nodes are the elements belonging to $E$.

**Definition 2.** ***(Distance-based generalisation operator)*** *Let $(X, d)$ be a connected metric space and let $\mathcal{L}$ be a pattern language. Given a generalisation operator $\Delta$, we will say that $\Delta$ is a (proper or hard) distance-based generalisation operator if, for every $E \subseteq X$, there exists a nerve $N(E)$ such that,*

- ***(proper)*** *For every pair of elements $x, y$ in $E$ such that they are directly linked in $N(E)$, $\Delta(E)$ includes **some** $I(X)$-path $P$ connecting $x$ and $y$ such that $d(x, y) = L(P)$.*
- ***(hard)*** *For every pair of elements $x, y$ in $E$ such that they are directly linked in $N(E)$, $\Delta(E)$ includes **all** $I(X)$-path $P$ connecting $x$ and $y$ such that $d(x, y) = L(P)$.*

[2] Here, the term connected refers to the well-known property for graphs.

Definition 2 can be difficult to understand. Let us see an example with a binary sets of elements $E = \{x, y\}$. In this case a proper distance-based operator will be that one computing a generalisation of $E$ which includes some of the paths built from the elements placed at the "middle" of $x$ and $y$. The generalisation is hard, if it includes all the paths built from the elements placed at the "middle" of $x$ and $y$. In what follows, we will refer to them as proper or hard operators. On the other hand, independently to the operator, we can say that a pattern generalises $E$ in a proper or a hard way if $Set(p)$ satisfies the conditions above (for further details see [5]).

The distinction between proper and hard is due to the fact that hard generalisations explains the distance among the elements better than the proper ones because it takes into account all the middle elements of two given elements and not only some of then as proper generalisations do. In fact, in some cases, proper generalisations do not always have an intuitive interpretation in terms of the distance involved, as we will see in next section.

## 4 Generalising set of graphs

In this section, we study some distance-based generalisation operators for graphs embedded in the metric spaces $(G, d_1)$ and $(G, d_2)$.

The process is as follows. First, for each metric space and using the pattern language $\mathcal{L}$ that will be further defined, we try to characterise which conditions a generalisation operator must satisfy to be a hard operator. From these hard operators we will study two important aspects: i) how expressive our pattern language is, and ii) how complicated the metric space is in order to compute patterns which inform about the distance between a group of elements. Finally, some patterns, leading to proper generalisations, will be examined as well.

The pattern language $\mathcal{L}$ we are working with is composed of two types of patterns: the first-kind ($\mathcal{L}_1$) and the second-kind ($\mathcal{L}_2$) patterns. The so-called first-kind patterns will be a set of graphs built from an alphabet of labels containing constant and variable symbols. Regarding the second-kind patterns, these are expressed in terms of the first ones and they are specially introduced to improve the expressiveness of ($\mathcal{L}_1$).

**Definition 3.** ***(First-kind patterns*** *($\mathcal{L}_1$)**)*** *Given $G$ the set of all the graphs over an alphabet of constant symbols $A$. If $X$ is a set of variable symbols such that for all $X_i \in X$, $X_i$ represents any constant symbol in $A$, then the language of first-kind patterns ($\mathcal{L}_1$) is defined as the set of all the graphs over the alphabet $A \cup X$.*

Roughly speaking, the first-kind pattern is the intensional representation of a set of graphs sharing a particular topological structure, just as we show in the following example.

*Example 2.* Given the first-kind pattern language ($\mathcal{L}_1$) defined from the set of constant symbols $A = \{a, b\}$ and the set of variable symbols $X = \{X_1, \ldots, X_n, \ldots\}$, consider the patterns $p_1$ in Figure 3.

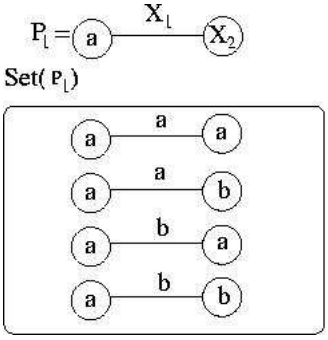


**Fig. 3.** A first-kind pattern representing a finite set of graphs.

This pattern represents only those graphs $g$ in $G$ made up of one edge and two vertices being one vertex labelled by the symbol $a$.

Although $\mathcal{L}_1$ permits quite interesting patterns, it still possesses some limitations. That is, imagine that we want to denote all the graphs in $G$ having a subgraph in common. Despite the fact that this request seems usual, there is no pattern in $\mathcal{L}_1$ expressing it. For this reason, the class of the second-kind patterns is introduced.

**Definition 4.** ***(Second-kind patterns $\mathcal{L}_2$)*** *Given the language of the the first-kind patterns $\mathcal{L}_1$, the language of the second-kind patterns $\mathcal{L}_2$ is defined as, $\mathcal{L}_2 = \{[p] : \forall p \in \mathcal{L}_1\} \cup \{\top\}$, where $[p]$ denotes all the graphs $g$ in $G$ having a subgraph covered by $p$ and $\top$ denotes the whole space $G$.*

*Example 3.* The second-kind pattern $p$ depicted in Figure 4 represents the set of all the graphs in $G$ containing the path[3] $a - a - b$.

Next, lest us define distance-based generalisation operators for $(G, d_1)$ and $(G, d_2)$ via $\mathcal{L}$.

### 4.1 Generalisation operators for $(G, d_1)$

Before defining a generalisation operator, we have proved that the metric space $(G, d_1)$ is a 1-connected metric space (see Proposition 1 in Appendix).

Now, let us characterise hard generalisation operators in $(G, d_1)$ using $\mathcal{L}$. Note that, for every finite set of graphs $\{g_i\}_{i=1}^{n>2}$ in $G$, an operator $\Delta(\{g_i\}_{i=1}^{n>2})$ will return a pattern belonging either to $\mathcal{L}_1$ or to $\mathcal{L}_2$. It can be shown that first-kind patterns do not represent connected sets (see Proposition 2 in Appendix) and therefore, according to Definition 1 they cannot represent a set computed by a generalisation operator. Then, the only possibility is that $\Delta$ computes a

[3] The concept of path referred here, is the well-known concept of path of a graph.

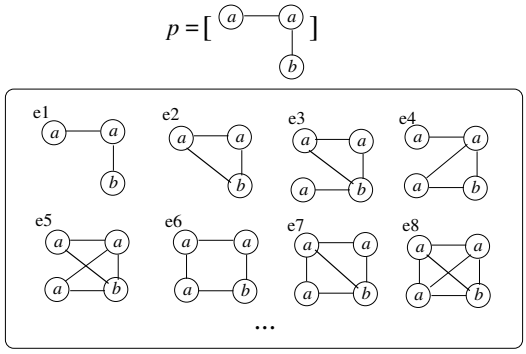


**Fig. 4.** An example of second-kind pattern using the notation $[\cdot]$.

second-kind pattern. Thus, a hard operator is given by (see Proposition 5 in Appendix):

$$\Delta(\{g_i\}_{i=1}^{n>2}) = \begin{cases} [p] & \text{if conditions (1) and (2) holds} \\ \top & \text{otherwise.} \end{cases}$$

where conditions (1) and (2) are:

**(1)** $p$ is a subgraph of the $mcs(\{r_i\}_{i=1}^{n})$, where $mcs(\{r_i\}_{i=1}^{n}) \neq \emptyset$ and each $r_i$ denotes one of the possible $vimcs(\{g_i\}_{i=n}^{n\geq 2})$.

**(2)** there exists a nerve $N(\{g_i\}_{i=n}^{n\geq 2})$ such that for every pair of graphs, $g_i$ and $g_j$, directly linked in the nerve $N$, all the possible $vimcs(g_i, g_j)$ are included in $\{r_i\}_{i=1}^{n}$.

As we can appreciate, the above conditions are extremely restrictive. For instance, regarding the two graphs depicted in the Figure 1, the pattern $[a-b]$ generalises $G_1$ and $G_2$ since they have the edge $a-b$ in common. However, this pattern is not hard because the common squared subgraph is a middle element of $G_1$ and $G_2$ but it is not covered by the pattern. In fact, $p$ does not satisfy condition (1) since $[a-b]$ is not a common subgraph of the $vicms(G_1, G_2)$. This fact gives us an idea about how difficult is computing hard operators in this space. Imaging an algorithm implementing $\Delta$, this would have to check if a subgraph of a set of graphs $G$ is in its turn a subgraph of the $vicms(G)$. According to this observation, the algorithms in the literature approaching the maximum common subgraph among a set of graphs [1] can not be used as an implementation of $\Delta$ because it can not be ensured that the returned subgraph belongs to the intersection of the *vicms*.

Another possibility is to use proper patterns, but in some cases it occurs that these do not explain the distance between two elements in a satisfactory way. This is shown in Figure 5. Both graphs have the subgraph $g_1$ in common, but by applying $d_1$, only one of the edges of $g_1$ is used to compute the distance between $g_1$ and $g_2$. Therefore, the pattern $[g_1]$ does not completely justify why the graphs are at this distance.

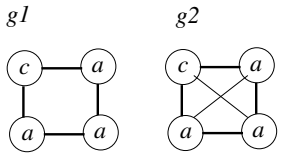


**Fig. 5.** Proper patterns in $(G, d_1)$.

According to the definition of $\Delta$, if the $mcs(r_{i\,i=1}^{n>2})$ is empty, the operator returns $\top$. This happens even when there exist subgroups of graphs in the set having some features in common (see left Figure 6). Although $\top$ is hard, it is not very useful because it is excessively general. This issue is treated with more detail in [5].

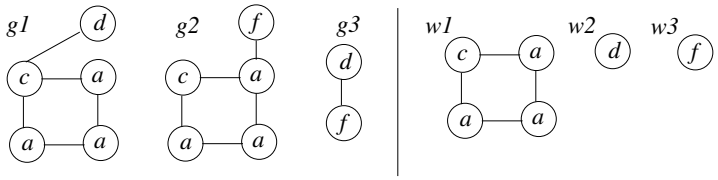


**Fig. 6.** A finite set of graphs belonging to $(G, d_1)$.

In Figure 6, every pair of graphs, $g_i$ and $g_j$ has a common subgraph $w_k$. But the pattern computed by a hard generalisation operator is $\top$. On the one hand, it is somehow consistent because there is no another pattern in $\mathcal{L}$ which generalise the three graphs. If the problem we are treating requires a more interesting solution, $\mathcal{L}$ should be improved. One possibility could be introduce a special symbol $+$ which permits to combine several patterns. That is, we would have new patterns such as $[p] + [q]$, which could be interpreted as $Set([p]) \cup Set([q])$. However, by doing that, two new hurdles arises. First, as the pattern language has changed, a new characterisation for the hard operators should be given. Secondly, since the size of $\mathcal{L}$ increases, it would take more time to the operator implementation to find the adequate pattern.

### 4.2 Generalisation operators for $(G, d_2)$

The metric space $(G, d_2)$ is 1-connected as well (see Proposition 1 in Appendix).

One of the advantages of working with this metric space is that hard generalisation operators can be characterised in a more natural way using $\mathcal{L}$. Doing a similar analysis that in the previous subsection, hard operators can be defined in $(G, d_2)$ as (see Proposition 6 in Appendix):

$$\Delta(g_{i\,i=1}^{n>2}) = \begin{cases} [p] & \text{if } p \text{ is a subgraph of the } mcs(\{g_i\}_{i=1}^{n\geq 2}) \\ \top & \text{otherwise.} \end{cases}$$

Unlike the space $(G, d_1)$, these hard operators could be implemented by using one of the several algorithms in the literature for searching common subgraphs among a set of graphs since now $\Delta$ directly uses the $mcs$ of the set of graphs instead of the $vicms$ of the set of graphs. Any subgraph returned by these algorithms can be perfectly used to define a hard generalisation.

However, the most general pattern $\top$ can be obtained in this scenario as well. It happens when $mcs(\{g_i\}_{i=1}^{n\geq 2}))$ is empty. As we know, to avoid that, we could introduce the operation $+$ over patterns defined as in the case above with the same inconveniences already mentioned.

Another alternative solution could be explored: to use patterns $[p]$ such that all the labels of $p$ are variables. For instance, in Figure 7, the graphs $g_1$ and $g_2$ do not have any subgraph in common, but both have the same topology. We could say, that $g_1$ and $g_2$ belong to the set of all the graphs having a squared subgraph structure represented by the pattern $[p]$. Note that this pattern belongs to $\mathcal{L}$. This pattern performs a proper generalisation. To show this it is enough to consider the 1-path in the metric space obtained in the following way. First, we start from $g_1$. Then, the rest of elements $g_i$ of the path are obtained by adding a vertex or an edge of $g_2$ to $g_{i-1}$. When this process ends we have a graph $g_1 - g_2$ which is the result of joining $g_2$ and $g_1$. Next, we remove iteratively the vertices and edges of $g_1$ from $g_1 - g_2$ until $g_2$ is obtained. The length of this 1-path (which is 8) is the same that $d_2(g_1, g_2)$ and it is included in $[p]$. Therefore, $[p]$ is a proper pattern. This is another example of proper pattern which does not explain why $g_1$ and $g_2$ are placed at this distance. Note that, $d_2$ counts only the number of different vertices and edges. For this reason, $d_2(g_1, g_3) = d_2(g_1, g_2) = 8$ but $g_3$ is not covered by $[p]$.

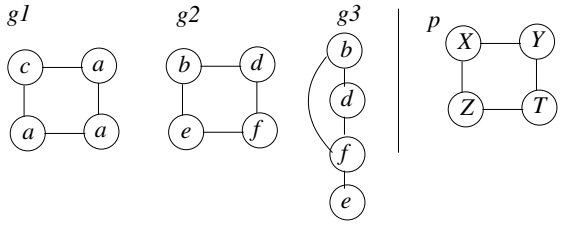


**Fig. 7.** A finite set of graphs belonging to $(G, d_2)$.

## 5 Conclusions and future work

Graph based learning is a challenging field due to the growing interest showed by several disciplines, such as biomedicine, in mining their source of data represented by means of graphs. However, most of the algorithms dealing with graphs,

specially distance-based, do not return a model using graph features (e.x. common paths, walks, etc.) explaining why a sample has been labelled or grouped in one way. Although this kind of models is useful from a comprehensibility point of view, obtaining them could lead to a inconsistence problems with the distance employed.

In this work, we use some of the concepts in [5] to analyse which consistent models can be obtained when graphs are embedded in metric spaces. According to our proposal, a model explaining a set of elements $E$ corresponds to a pattern belonging to a pattern language. We say that a model/pattern for $E$ is consistent when it is computed by a special generalisation operator, the so-called hard or proper generalisation operators. In this paper, we have mainly focused on characterising hard operators, although proper ones have been commented as well, for the metric spaces $(G, d_1)$ and $(G, d_2)$. The distance $d_1$ is found in [2] and $d_2$ is a variation of $d_1$ proposed by us. Hard operators are interesting because they computes patterns more consistent to the distance than proper operators do. Additionally, as we have seen, hard operators can bring information concerning how complex the metric space is or how expressive the pattern language is.

As future work, we are investigating on several topics, some of them have been slightly mentioned here, not only for graphs but also anther sort of data. First, we are developing distance-based cost functions to quantify how complex and general a pattern is for a given set of samples. This function will let us define operators computing patterns which reach a trade-off between overfitting and simplicity. Then, we aim to develop distance-based operators using pattern languages as expressive as regular languages. In this line, we think that some of the grammar inference algorithms could be upgraded for this purpose.

## References


1. E. Bengoetxea. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. PhD thesis, 2003.
2. H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.
3. S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer-Verlag, Berlin, September 2001.
4. V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Distance based generalisation. In *Proc. of the 15th International Conference on Inductive Logic Programming, ILP*, volume 3625 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2005.
5. V. Estruch, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. On the relationship between distance and generalisation. Technical report, Departamento de Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia, http://www.dsic.upv.es/$\tilde{}$flip/#Papers, 2006.

## 6 Appendix

**Proposition 1.** *The metric space* $(G, d_i)$ $(i = 1, 2)$ *is a* 1*-connected metric space.*

*Proof.* The proof is direct for both metric spaces. Focusing on $(G, d_1)$, we will see that for every pair of graphs $g_1$ and $g_2$ in $G$, there exists a 1-path connecting both. Observe that, when a vertex is removed from $g_i$ $(i = 1, 2)$, the new graph obtained $g_i'$ is a subgraph of $g_i$ and obviously $d_1(g_i, g_i') = 1$. Applying this operation (vertex removing) over $g_i'$ iteratively, we obtain two 1-paths, $P_1$ and $P_2$, connecting $g_1$ and $g_2$ respectively, to the empty graph. That is,

$$P_1 \equiv g_1 \to g_1' \to \cdots \to \emptyset \text{ and } P_2 \equiv g_2 \to g_2' \to \cdots \to \emptyset$$

Finally, joining $P_1$ and $P_2$ using the empty graph as common element, the 1-path connecting $g_1$ and $g_2$ is obtained.

As for $(G, d_2)$, the demonstration is exactly the same that the latter one. For this reason, we will simply sketch it. Given two graphs $g_1$ and $g_2$ belonging to $G$, there are always two 1-paths, $P_1$ and $P_2$, connecting $g_1$ and $g_2$ to the empty graph. They are obtained by iteratively removing vertices and edges from $g_i$ $(i = 1, 2)$. Finally, joining the paths $P_1$ and $P_2$ using the empty graph as common element, we have the 1-path connecting $g_1$ and $g_2$.

**Proposition 2.** *Given the metric space* $(G, d_i)$ $(i = 1, 2)$ *and the pattern language* $\mathcal{L}$*, then for every pattern* $p \in \mathcal{L}_1$ *where* $p$ *contains variables,* $Set(p)$ *is not a connected set in* $(G, d_i)$*.*

*Proof.* We will proceed by contradiction. Let us suppose that there exist two graphs $g$ and $g'$ in $Set(p)$ such that $d_i(g, g') = 1$. According to how both distances are defined, if $d_i(g, g') = 1$, we necessarily have that $g' \subset g$. But if it happens, $g'$ is not an instance of the pattern $p$, and this is impossible. Therefore, for every $g$ and $g'$ in $Set(p)$, $d_i(g, g') > 1$, and i.e., $Set(p)$ is not a connect set.

**Proposition 3.** *Given the metric space* $(G, d_i)$ $(i = 1, 2)$ *and the pattern language* $\mathcal{L}$*, then for every pattern* $p \in \mathcal{L}_1$*,* $Set([p])$ *is a connected set in* $(G, d_i)$*.*

*Proof.* Although the proof is straightforward, for the sake of comprehensibility, it will be organised in two parts. First, the specific case when $p$ does not contain variables will be addressed. Then, the general case, that is, when $p$ contains variables, will be proved by reducing it to the first one.

1. The first-kind pattern $p$ does not contain variables: Let us see that for every pair of graphs $g_1$ and $g_2$ in $Set([p])$, they are connected by 1-path included in $Set(p)$. For this purpose, we introduce an artificial graph called $g_1 - g_2$. This graph is obtained by linking one vertex $v_1$ in $g_1$ and one vertex $v_2$ in $g_2$ by an edge. Clearly, the graph $g_1 - g_2$ belongs to $Set([p])$ because $g_1 - g_2$ is an instance of the pattern $p$. Additionally, a 1-path connecting $g_1$ to $g_1 - g_2$ and another one connecting $g_2$ to $g_1 - g_2$ (both included in $Set([p])$) can be obtained. Hence, the graphs $g_1$ and $g_2$ are connected by means of a 1-path included in $Set([p])$).

2. The first-kind pattern $p$ contains variables: In this case, we know that $Set(p) = \{p_1, \ldots, p_n\}$ where each $p_i$ is a possible instance of $p$. Then, we can write:

   $$Set([p]) = \cup_{i=1}^{n} Set([p_i])$$

   Now, for every two graphs $g'$ and $g$ in $Set([p])$, we will construct a 1-path, included in $Set([p])$, connecting them. Naturally, there exist two patterns $p_i$ and $p_j$ in $Set(p)$ such that $g \in Set([p_i])$ and $g' \in Set[p_j]$. If $p_i = p_j$, then $g$ and $g'$ are connected by means of a 1-path since we have just seen that $Set([p_i])$ is a connected set. If not, the proof is not much more complicated. Note that $Set([p_i]) \cap Set[p_j]$ is not empty. This is immediate because the graph $p_i - p_j$ belongs to both $Set([p_i])$ and $Set([p_j])$. Then, we know that $g$ can be connected to $p_i - p_j$ by means of a 1-path in $Set[p_i]$, and the same for $g'$ in $Set([p_j])$. Thus, we can define the following 1-path connecting $g$ and $g'$:

   $$g \rightarrow \ldots \rightarrow p_1 - p_2 \rightarrow \ldots \rightarrow g'$$

   Of course, this path is included in $Set([p])$, and i.e., $Set([p])$ is a connected set.

Finally, we can affirm that for every pattern $p \in \mathcal{L}_2$, $Set([p])$ is a connected set.

From this latter Proposition, we can conclude that only the second-kind patterns can potentially be used by a generalisation operator defined both in $(G, d_1)$ and $(G, d_2)$.

**Proposition 4.** *Let $g_1$,$g_2$ and $g_3$ be three graphs belonging to $(G, d_i)$ $(i = 1, 2)$ such that $mcs(g_1, g_2) \neq \emptyset$, if $d_i(g_1, g_2) = d_i(g_1, g_3) + d_i(g_3, g_2)$ then $g_3$ contains:*

- *Considering the space $(G, d_1)$, at least one of the vertex-induced $mcs(g_1, g_2)$*
- *Considering the space $(G, d_2)$, the $mcs(g_1, g_2)$.*

*Proof.* Considering the space $(G, d_1)$: According to [] (referencia Bunke paper), we can express the equality $d_1(g_1, g_2) = d_1(g_1, g_3) + d_1(g_3, g_2)$ in terms of the set of vertices of the involved graphs:

$$|V_{g_1}| + |V_{g_2}| - 2|V_{g_{1,2}}| = |V_{g_1}| + |V_{g_3}| - 2|V_{g_{1,3}}| + |V_{g_2}| + |V_{g_3}| - 2|V_{g_{2,3}}| \ (1) \leftrightarrow$$

$$-|V_{g_{1,2}}| = |V_{g_3}| - |V_{g_{1,3}}| - |V_{g_{2,3}}| \ (2),$$

where $V_{g_{i,j}}$ denotes the set of vertices correspondent to the vertex-induced $mcs(g_i, g_j)$. Now, let us see that the vertex-induced $mcs(g_{1,3}, g_{2,3}) \neq \emptyset$. If it was the emtpy graph, we would have that

$$|V_3| - |V_{1,3}| - |V_{2,3}| \geq 0 \ (3)$$

and (2) would not hold. Hence, $g_{1,3}$ and $g_{2,3}$ have a vertex-induced subgraph in common. In fact, it is immediate that the vertex-induced $mcs(g_{1,3}, g_{2,3}) \subset g_{1,2}$.

Let us decompose the set of vertices $V_{g_{1,3}}$ (respectively $V_{g_{2,3}}$) into two disjoint sets $X_{g_{1,3}}$ ($X_{g_{2,3}}$) and $Y$, where $X_{g_{1,3}}$ contains those vertices of $g_{1,3}$ not belonging

to the vertex-induced $mcs(g_{1,3}, g_{2,3})$ and $Y$ precisely contains the vertices of the vertex-induced $mcs(g_{1,3}, g_{2,3})$. Thus, we can rewrite (2) as

$$-|V_{g_{1,2}}| = |V_{g_3}| - |X_{g_{1,3}}| - |X_{g_{2,3}}| - 2|Y| \quad (4)$$

In principle, as the vertex-induced $mcs(g_{1,3}, g_{2,3}) \subset g_{1,2}$, we can affirm that $|Y| \leq |V_{g_{1,2}}|$. But as (4) holds, the following inequality

$$|V_{g_3}| - |X_{g_{1,3}}| - |X_{g_{2,3}}| - |Y| \geq 0 \quad (5)$$

necessarily vanishes and $|Y| = |V_{g_{1,2}}|$. This implies that the vertex-induced $mcs(g_{1,3}, g_{2,3}) = g_{1,2}$ and finally, that $g_3$ contains the vertex-induced $mcs(g_1, g_2)$. Considering the metric space $(G, d_2)$: The demonstration is quite similar to the latter one. In this case, the distance $d_2$ between two graphs, can be calculated taking their set of vertices and edges into account. In this way, the expression $d_2(g_1, g_2) = d_2(g_1, g_3) + d_2(g_3, g_2)$ can be written as

$$|V_{g_1}| + |V_{g_2}| - 2|V_{g_{1,2}}| + |E_{g_1}| + |E_{g_2}| - 2|E_{g_{1,2}}| =$$

$$|V_{g_1}| + |V_{g_3}| - 2|V_{g_{1,3}}| + |E_{g_1}| + |E_{g_3}| - 2|E_{g_{1,3}}| +$$

$$|V_{g_2}| + |V_{g_3}| - 2|V_{g_{2,3}}| + |E_{g_2}| + |E_{g_3}| - 2|E_{g_{2,3}}| \quad (6)$$

and simplifying,

$$-|V_{g_{1,2}}| - |E_{g_{1,2}}| = |V_{g_3}| - |V_{g_{1,3}}| - |V_{g_{2,3}} + |E_{g_3}| - |E_{g_{1,3}}| - |E_{g_{2,3}}| \quad (7)$$

where $V_{g_{i,j}}$ and $E_{g_{i,j}}$ denotes the set of vertices correspondent to the $mcs(g_i, g_j)$.

Again, if (7) holds, necessarily $mcs(g_{1,3}, g_{2,3}) \neq \emptyset$. In fact, $mcs(g_{1,3}, g_{2,3}) \subset g_{1,2}$. Let us decompose (7) into two new equations:

$$-|V_{g_{1,2}}| = |V_{g_3}| - |V_{g_{1,3}}| - |V_{g_{2,3}}| \quad (8)$$

$$-|E_{g_{1,2}}| = |E_{g_3}| - |E_{g_{1,3}}| - |E_{g_{2,3}}| \quad (9)$$

Note that (8) and (9) are structurally identical to (2). The same reasoning, that we did in the case above, over equations (8) and (9) lead us to affirm that the number of vertices and edges of $mcs(g_{1,3}, g_{2,3})$ is equal to $|V_{g_{1,2}}|$ and $|E_{g_{1,2}}|$, respectively. Hence, $mcs(g_{1,3}, g_{2,3}) = g_{1,2}$ and we can say that $g_3$ contains the $mcs(g_1, g_2)$.

**Proposition 5.** *Let $\{g_i\}_{i=n}^{n\geq 2}$ be a set of graphs belonging to $G$ is not empty. Given the metric space $(G, d_1)$ and the mapping $\Delta(\{g_i\}_{i=n}^{n\geq 2}) \rightarrow \mathcal{L}_2$, we will say that $\Delta(\{g_i\}_{i=n}^{n\geq 2}) = [p]$ is a hard-proper generalisation if:*

***(1)*** *$p$ is a subgraph of the $mcs(\{r_i\}_{i=1}^{n}) \neq \emptyset$, where each $r_i$ denotes one of the possible vertex-induced $mcs(\{g_i\}_{i=n}^{n\geq 2})$.*

***(2)*** *there exists a nerve $N(\{g_i\}_{i=n}^{n\geq 2})$ such that for every pair of related graphs, $g_i$ and $g_j$, in the nerve $N$, all the possible vertex-induced $mcs(g_i, g_j)$ are included in $\{r_i\}_{i=1}^{n}$.*

*Proof.* As $[p]$ belongs to $\mathcal{L}_2$, $[p]$ is a connected set. Then, as for every pair of graphs $g_i$ and $g_j$ directly linked in the graph, all its vertex-induced $mcs(g_i, g_j)$ are included in $\{r_i\}_{i=1}^{n}$ and $p$ is a subgraph of $mcs(\{r_i\}_{i=1}^{n})$, we can ensure that all the possible vertex-induced $mcs(g_i, g_j)$ is in $[p]$. Next, using Proposition 4, if a graph $g$ satisfies that $d_1(g_i, g_j) = d_1(g_1, g) + d_1(g_j, g)$, $g$ contains a vertex-induced $mcs(g_i, g_j)$. Therefore, $g$ belongs to $[p]$ and $\Delta$ is a locally hard proper generalisation.

**Proposition 6.** *Let $\{g_i\}_{i=1}^{n\geq 2}$ be a finite set of graphs belonging to $G$ such that $mcs(\{g_i\}_{i=1}^{n\geq 2})$ is not empty. Given the metric space $(G, d_2)$ and the mapping $\Delta(\{g_i\}_{i=1}^{n\geq 2}) \rightarrow \mathcal{L}_2$, we will say that $\Delta(\{g_i\}_{i=1}^{n\geq 2}) = [p]$ is a hard-proper generalisation if $p$ is a subgraph of the $mcs(\{g_i\}_{i=1}^{n\geq 2})$.*

*Proof.* Again, as $[p]$ belongs to $\mathcal{L}_2$, $[p]$ is a connected set. Now, using Proposition 4, if a graph $g$ satisfies that $d_2(g_i, g_j) = d_2(g_1, g) + d_2(g_j, g)$, $g$ contains the $mcs(g_i, g_j)$. As $p$ is a subgraph of $mcs(\{g_i\}_{i=1}^{n\geq 2})$, $[p]$ necessarily covers $g$, and i.e., $\Delta$ is a locally hard-proper generalisation.