Pattern Analysis and Applications (2024) 27:143
https://doi.org/10.1007/s10044-024-01366-w

THEORETICAL ADVANCES

# Graphfingerprint: graph embedding of graphs with almost constant sub-structures

**Francesc Serratosa**[1]

Received: 24 April 2023 / Accepted: 21 October 2024 / Published online: 14 November 2024


**Abstract**
In some machine learning applications, graphs tend to be composed of a large number of tiny almost constant sub-structures. The current embedding methods are not prepared for this type of graphs and thus, their representational power tends to be very low. Our aim is to define a new graph embedding that considers this specific type of graphs. We present GraphFingerprint, which is a new embedding method that specifically considers the fact that graphs are composed of millions of almost constant sub-structures. The three-dimensional characterisation of a chemical metal-oxide nanocompound easily fits in these types of graphs, which nodes are atoms and edges are their bonds. Our graph embedding method has been used to predict the toxicity of these nanocompounds, achieving a high accuracy compared to other embedding methods. The representational power of the current embedding methods do not properly satisfy the requirements of some machine learning applications based on graphs, for this reason, a new embedding method has been defined and heuristically demonstrated that achieves good accuracy.

**Keywords** Graph embedding · Graph regression · Graph classification · Metal-oxide nanocompound · Chemical 3D-structure · GraphFingerprint · NanoFingerprint

## 1 Introduction

An attributed graph, in general, is a data structure depicting a collection of entities represented as nodes, and their pairwise relationships represented as edges. Attributed graphs have been used for pattern recognition and machine learning for several decades to represent objects [1, 2].

In biotechnology, there is a growing interest in having graph-based techniques applied to machine learning. They are used to represent chemical compounds and then predict their toxicity [3]. This can be attributed to their effectiveness in characterising instances of data with complex structures and rich attributes and also, the goodness of distances between graphs, such as the Graph edit distance [4, 5], to capture the dissimilarity between graphs. Nevertheless, this classical framework has the main drawback that computing a graph distance has usually a high computational cost (although the existence of sub-optimal algorithms). This is an important issue since the computational time for learning the model and also for using it could be inadmissible. This is because some chemical compounds could be composed of thousands of atoms and chemical databases (for learning and testing) could have millions of compounds.

To overcome this problem, some researches have applied Graph Embedding, Graph Convolutional Networks or Graph Autoencoders to classify or to predict some properties of chemical compounds [6, 7]. These methods learn the local structures and semantics of the attributed graphs and convert them into a latent space, represented as a matrix or vector of Real numbers. Then, given this space, classical machine learning methods can be applied. Note it is key for these methods that not only the involved graphs have to be different (large graph distance) but also having a rich internal variability (composed of several different local structures and attributes). Unfortunately, there are some pattern recognition applications in which elements are represented by graphs that their internal variability is very low.

Chemical metal-oxide nanocompounds are chemical compounds composed by only tens of thousands of atoms and they only have two different types of atoms, oxygen and

✉ Francesc Serratosa
francesc.serratosa@urv.cat

[1] Department d'Informàtica i Matemàtiques, Universitat Rovira i Virgili, 43007 Tarragona, Catalonia, Spain

a metal. Moreover, their internal sub-structures are almost constant. Currently, there are several models to predict their properties, such as the toxicity, which are based on global properties but they do not use the information of their three-dimensional structure. Supported by the European projects NanoInformatix[1] and Sbd4nano[2] we modelled, for the first time, the toxicity of metal-oxide nanocompounds based on their three-dimensional structure. To do so, we represented these structures by attributed graphs but it was not possible to use the methods commented in the previous paragraphs due to graphs were composed of almost constant sub-structures and only to types of attributes appeared in the nodes. Thus, graphs had not enough internal variability.

Considering this feature, we initially represented the nanocompound by an attributed graph and then embedded it as a vector with a model based on counting the graphlets of the graph [8]. We thought that a representation based on counting the local structures could properly match the small variability of the graphs that represents the nanocompound. Nevertheless, we realised that some nodes (atoms) had a large number of edges (bonds), and also, some nodes (atoms) had a very small number of edges (bonds). Then, if we wanted the embedding to take into consideration all combinations of nodes, the embedding vector would have to be huge, and it was not acceptable in our application.

Moreover, we also wanted to compare or method to state of the art graph transformers [9]. Note this method includes the node position into the attention mechanism with the aim of distinguishing the position of each node. Nevertheless, due to the graphs are almost constant, the incorporated information relative to the position seems to be almost constant throughout all the nodes in the graphs, thus, adding little information to the system.

The aim of this paper is to present a new graph embedding specifically designed for applications of graph regression or graph classification in which attributed graphs are composed of almost constant sub-structures and attributes are almost the same. We have called *GraphFingerprint* to this graph embedding and it has been used for metal-oxide nanocompound toxicity prediction. In the next section, we recapitulate the old graph embedding and graph regression methods, and also we comment the new graph convolutional and graph autoencoder methods. In Sect. 3, we explain in detail our embedding method and in Sect. 4, we explain the application of GraphFingerprint in nanocompound toxicity prediction. We show that our method achieves better accuracy compared to current graph regression and classification methods. Section 5 and Sect. 6 concludes this paper and presents future work, respectively.

[1] https://www.nanoinformatix.eu

[2] https://www.sbd4nano.eu

## 2 Graphs and graph embedding

Graphs are effective mathematical tools for characterising instances of data with complex structures and rich attributes. For this reason they have been used for automatically classifying objects in pattern recognition or deducing global properties through regression methods for almost fifty years [1, 2].

The classical K-Nearest Neighbours algorithm could be used for graph classification or graph regression purposes. In this case, a distance between graphs have to be used, such as the graph edit distance [4, 5]. Nevertheless, one downside of these methods is the calculation of graph distances [10–12], which would increase the computational time in correspondence to the number of data elements.

With the aim of avoiding the calculation of graph distances, it is usual to move from the graph domain into a Euclidean domain. This conversion is achieved through graph embedding techniques. Some of these techniques were presented some years ago, in which the embedding process did not use trainable parameters [13]. More recently, Graph Convolutional Networks [14, 15] and Graph Autoencoders [7, 16, 17] have emerged as tools to embed graphs. The main difference of these last methods and the old embedding methods is the ability to select the intrinsic features of the graphs through learned parameters.

## 3 Method: GraphFingerprint for graph embedding

GraphFingerprint is an attributed graph embedding defined as a vector of numbers, which has been designed to encode attributed graphs composed of almost constant small sub-structures. Moreover, nodes are labelled in only two classes and edges are unattributed. This type of graphs appear in some specific and completely different scenarios, such as toxicity prediction of chemical nanocompounds (graphs represent the three-dimensional structure of the compound) or social networks analysis (for instance, nodes are people labelled by their voting tendencies).

More formally. We define a graph $G = (V, E)$ as a set of nodes $V$ and a set of edges $E$. $G_i$ is the $i$ th node in $V$ and $G_{i,j}$ is the edge in $E$ between the $i$ th node and the $j$ th node in $V$. Moreover, $\gamma_i = \{N_k, N_p\}$ is the label or attribute of node $G_i$. $N_t$ is a cardinal attribute, being, $1 \leq t \leq T$. Note given a specific graphs, it can only have two types of attributes, $N_k$ and $N_p$ and it is not possible to have three different types of attributes, such as, $N_k$, $N_p$ and $N_n$, being $k \neq p \neq n$. For this reason, since now, we classify the nodes of a specific graph in two types: *O* and *M*, where class *O* are

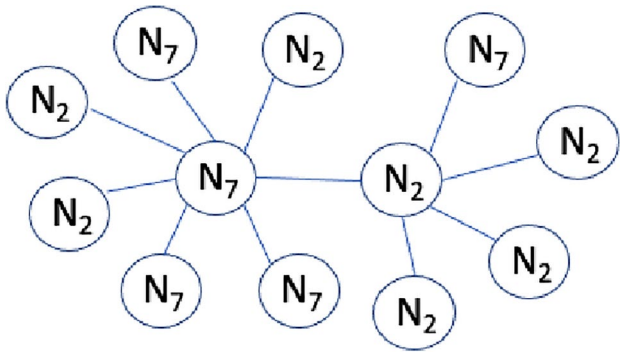


**Fig. 1** A graph composed of twelve nodes. In this case, class *O* represents nodes with attribute $N_2$ and class *M* represents nodes with attribute $N_7$

**Table 1** Local structures with their acronyms

| Position | Global features |
|---|---|
| $O(x)$ | It represents a node type *O* that have *x* edges |
| $M(x)$ | It represents any node of type *M* that have *x* edges |
| $O(x, y)$ | A local structure composed of a central node of type *O* connected to x nodes of type *O* and y nodes of type of *M*. Note these x *O* and y *M* could be connected to other nodes |
| $M(x, y)$ | Similarly, a local structure composed of a central node of type *M* connected to x nodes of type *O* and y nodes of type *M* |
| O(x,y)-O(x′,y′) | A structure that is composed of two of the previous ones. It is composed of an $O(x, y)$ and an $O(x^{'}, y^{'})$ whose central *O* are connected by an edge |
| M(x,y)-M(x′,y′) | In a similar way, it is composed of an $M(x, y)$ and an $M(x^{'}, y^{'})$ whose central *M* are connected by an edge |
| O(x,y)-M(x′,y′) | Finally, a similar structure but that connects an $O(x, y)$ and an $M(x^{'}, y^{'})$ whose central nodes are connected |

the nodes that have attribute $N_k$ and class *M* are the nodes that have attribute $N_p$, being $k < p$.

Figure 1 shows a naive example of a graph composed of twelve nodes, seven of them have attribute $N_2$ and five of them have attribute $N_7$. In this graph, class *O* represents nodes with attribute $N_2$ and class *M* represents nodes with attribute $N_7$.

The rest of this section is composed of two subsections. In the first one, we define the local substructures and in the second one we define the GraphFingerprint.

### 3.1 Local substructures

A *local structure* is a small set of connected nodes in the graph. In this section, we present some *local structures* that are used to define, in an organised manner, the GraphFingerprint. They are summarised in Table 1.

**Table 2** Section 1: global information

| Position | Global features |
|---|---|
| 1 | *MAX* (Maximum number of edges per node) |
| 2 | $N_k$ (the attribute of nodes of type *O*) |
| 3 | $N_p$ (the attribute of nodes of type *M*) |
| 4 | Number of nodes type *O* in the graph |
| 5 | Number of nodes type *M* in the graph |

**Table 3** Section 2: Node information

| Position | Appearance of the node features |
|---|---|
| 1 | *O*(1) |
| .. | .. |
| *MAX* | *O*(*MAX*) |
| $MAX + 1$ | *M*(1) |
| .. | .. |
| 2*MAX* | *M*(*MAX*) |

Note graph in Fig. 1 shows a reproduction of the local structure $O(2, 3) - M(3, 4)$, where class *O* represents nodes with attribute $N_2$ and class *M* represents nodes with attribute $N_7$.

### 3.2 GraphFingerprint definition

A GraphFingerprint is a vector of Natural numbers that counts appearances of *local structures*. It is split up in four main sections: Global information, Node information, Edge information and Structural information. Note the number of nodes and edges involved in each local structure increases in each section, that is, in each section, larger local structures are considered.

In the rest of this section, we describe what is stored in each position of the vector that represents the GraphFingerprint. For instance, the line "$(MAX + 1)^2 + MAX + 1$ : Number of *M*(0, *MAX*)" in Section 3 represents that the value that appears at position $(MAX + 1)^2 + MAX + 1$ of Section 3 is the number of appearances of the local structure *M*(0, *MAX*).

- **Section 1 : global information** The first section only specifies the two attributes of the nodes and counts the appearances of the two types of nodes. Moreover, it imposes a maximum order of nodes: MAX. Nodes that have larger number are not considered. This value is a parameter in the construction of GraphFingerprint and it is useful to impose the length of the GraphFingerprint to be constant in a specific database or application. Table 2 details this section of the embedding.
- **Section 2 : Node information**

Section 2 is composed of $2 * MAX$ positions that count the number of appearances of nodes given their attribute (O or M) and number of edges. Table 3 details this section of the embedding.

- **Section 3 : Edge information**

Section 3 includes the information of the local structures $O(x, y)$ and $M(x, y)$. It is composed of $2(MAX+1)^2$ positions that count the number of appearances of nodes given their attribute (O or M) and also the number of edges they have connected to nodes type O or M. Table 4 details this section of the embedding.

- **Section 4 : Structural information**

Section 4 includes the information of the local structures $O(x,y)-O(x^{'},y^{'})$, $M(x,y)-M(x^{'},y^{'})$ and $O(x,y)-M(x^{'},y^{'})$. It is composed of $3(MAX+1)^4$ positions that count the largest considered local structure. As previously commented, Fig. 1 is an example of one of these structures. It shows a reproduction of the local structure $O(2,3)-M(3,4)$. Table 5 details this section of the embedding.

The length of the GraphFingerprint depends on the maximum accepted node order MAX, length($\text{GraphFingerprint}_{\text{MAX}}$)= $5+2\text{MAX}+2(\text{MAX}+1)^2+3(\text{MAX}+1)^4$. Thus, the theoretical worst computational cost of generating GraphFingerprints is in fourth power of the maximum order of nodes (MAX).

## 4 Experimental section

Modelling size-realistic nanomaterials to analyse some their properties, such as toxicity, solubility, or electronic structure, is a current challenge in computational and theoretical chemistry. The representation of all-atom three-dimensional structure of the nanocompound would be ideal, as it could account explicitly for structural effects. However, the use of the whole structure is tedious due to the huge data management and the structural complexity that accompanies the chemical nanocompound. Developing appropriate tools that allow the quantitative analysis of the three-dimensional structure, as well as the selection of regions of interest such as the most external atoms, is a crucial step to enable efficient analysis and processing of model nanostructures.

GraphFingerprints have been applied to embed chemical nanocompounds for their post processing and predicting their toxicity. Specifically, they have been applied to metal-oxide nanocompounds, which are chemical compounds that have the following three features: (1) They are composed of only some thousands of atoms. (2) They have only two types of atoms, the ones that are oxygen and the other ones that are metals. (3) Their local substructures are almost constant. Thus, we performed nanocompound toxicity prediction through graph embedding and regression.

**Table 4** Section 3: Edge information

| Position | Appearance of the edge features |
|---|---|
| 1 | $O(0, 0)$ |
| .. | .. |
| $MAX+1$ | $O(0, MAX)$ |
| $MAX+2$ | $O(1, 0)$ |
| .. | .. |
| $2(MAX+1)$ | $O(1, MAX)$ |
| .. | .. |
| $(MAX+1)^2$ | $O(MAX, MAX)$ |
| $(MAX+1)^2+1$ | $M(0, 0)$ |
| .. | .. |
| $(MAX+1)^2+MAX+1$ | $M(0, MAX)$ |
| $(MAX+1)^2+MAX+2$ | $M(1, 2)$ |
| .. | .. |
| $(MAX+1)^2+2(MAX+1)$ | $M(1, MAX)$ |
| .. | .. |
| $(MAX+1)^2+(MAX+1)^2$ | $M(MAX, MAX)$ |

**Table 5** Section 4: Structural information

| Position | Appearance of local structures |
|---|---|
| 1 | O(0,0) – O(0,0) |
| .. | .. |
| $(MAX+1)^2$ | O(0,MAX) – O(0,MAX) |
| $(MAX+1)^2+1$ | O(1,0) – O(1,0) |
| .. | .. |
| $(MAX+1)^4$ | O(MAX,MAX) – O(MAX,MAX) |
| $(MAX+1)^4+1$ | M(0,0) – M(0,0) |
| .. | .. |
| $(MAX+1)^4+(MAX+1)^4$ | M(MAX,MAX) – M(MAX,MAX) |
| $(\text{MAX}+1)^4+(\text{MAX}+1)^4+1$ | O(0,0) – M(0,0) |
| .. | .. |
| $(\text{MAX}+1)^4+(\text{MAX}+1)^4+(\text{MAX}+1)^4$ | O(MAX,MAX) – O(MAX,MAX) |

The following sections explain the process of embedding a nanocompound into a GraphFigerprint and the regression applied to GraphFingerprint to deduce their toxicity. Moreover, we also depict a web site from which users can generate GraphFingerprints, compute nanocompound toxicity predictions or other graph-related operations.

### 4.1 From metal-oxide nanocompound to GraphFingerprint

Metal-oxide nanocompounds are chemical compounds composed of two types of atoms, oxygen and any type of metal, for instance, Al, Cu, Fe, Ti or Zn. The most

common metal-oxide nanoparticles are $Al_2O_3$, CuO, $Fe_2O_3$, $TiO_2$, $SiO_2$ and ZnO. The analysis and prediction of their reactivity and toxicity is crucial for better understanding these compounds and their use in the industry [18]. Given that experimental evaluation of the safety of chemicals is expensive and time-consuming, computational methods have been found to be efficient alternatives to predict their toxicity level.

The three-dimensional structure of chemical nanocopounds can be represented as attributed graphs [19]. Attributed graphs have been applied in machine learning for chemical compound toxicity prediction for a long time [20, 21]. In our case, atoms in the nanocompound are represented by nodes and chemical bounds by edges. Due to the specific simplicity of metal-oxide nanocompounds, nodes can be attached with the type of atom (O, Al, Cu, Fe, Ti or Zn) and edges do not have attributes since all bonds tend to be of the same non-covalent type. Specifically, and considering the graph definition in Sect. 3, we define $\gamma_i = \{\mathrm{O}\}$ for the oxygen atoms and $\gamma_i = \{Al, Cu, Fe, Ti, Zn\}$ for the metal ones. Then, class *O* atoms are the ones that $\gamma_i = \{\mathrm{O}\}$ and class *M* atoms are the ones that $\gamma_i = \{Al, Cu, Fe, Ti, Zn\}$. Note the three-dimensional position of the atoms is not represented in the graph.

As an example, Fig. 2 shows the chemical nanocompound $Al_2O_3$ that has size 13 Angstroms. It is only composed of 30 atoms, 18 of them are oxygen (in red) and 12 of them are aluminium (in grey). We can draw this compound since we have its three-dimensional structure.[3]

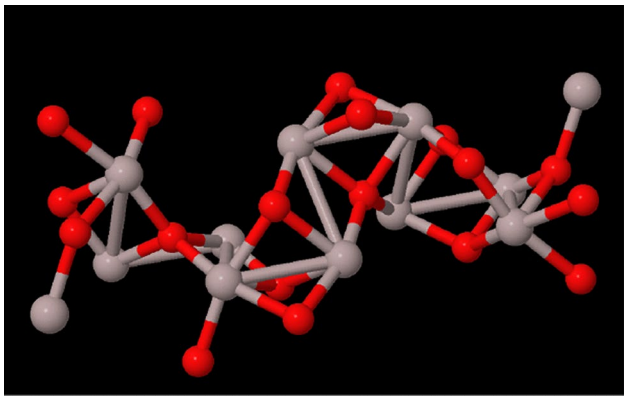

**Fig. 2** Chemical nanocompound $Al_2O_3$ of size 13 Angstroms. Oxygen atoms in red and aluminium atoms in grey

Figure 3 shows the graph generated by the chemical nanocompound $Al_2O_3$ shown in Fig. 2. Figure 4 shows its GraphFingerprint. Since GraphFingerprint are usually very sparse, we show only the non-empty elements. The first number represents the position of the element in the vector.

[3] The usual file format to describe the three-dimensional structure of nanocompounds is '.XYZ'. It is a well-know and simple format that, for each atom, it keeps the information of the type of atom and its position (x,y,z). No information about the bonds is included and thus, it has to be deduced through chemical-physical properties.

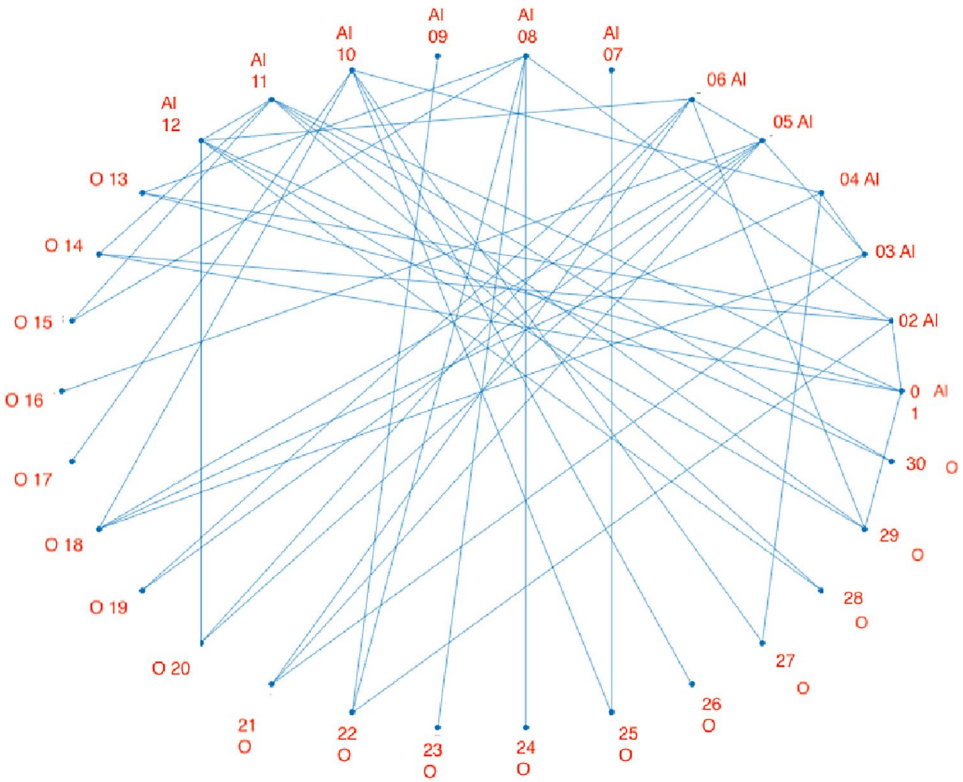


**Fig. 3** Graph that represents the chemical nanocompound $Al_2O_3$ shown in Fig. 2. The first 12 nodes are Aluminium atoms and the rest of them are oxygen atoms. The order of appearance is not related to the three-dimensional position since the graph do not have this information

```
Section 1
max: 10
attr(O): 8
attr(M): 13
num(O): 18
num(M): 12
Section2
5-> O[1]: 5
6-> O[2]: 6
7-> O[3]: 5
8-> O[4]: 2
16-> M[1]: 2
19-> M[4]: 2
20-> M[5]: 2
21-> M[6]: 4
22-> M[7]: 2

Section 3
27-> O[0,1]: 5
28-> O[0,2]: 6
29-> O[0,3]: 5
30-> O[0,4]: 2
158-> M[1,0]: 2
171-> M[2,2]: 2
182-> M[3,2]: 2
193-> M[4,2]: 2
203-> M[5,1]: 2
204-> M[5,2]: 2
```

```
Section4
17841-> M[2,2]_M[2,2]: 1
19183-> M[3,2]_M[3,2]: 1
20525-> M[4,2]_M[4,2]: 1
21713-> M[5,1]_M[2,2]: 1
21724-> M[5,1]_M[3,2]: 1
21834-> M[5,2]_M[2,2]: 1
21845-> M[5,2]_M[3,2]: 1
21856-> M[5,2]_M[4,2]: 2
29732-> O[0,1]_M[5,1]: 4
29733-> O[0,1]_M[5,2]: 1
29808-> O[0,2]_M[1,0]: 1
29821-> O[0,2]_M[2,2]: 1
29843-> O[0,2]_M[4,2]: 3
29853-> O[0,2]_M[5,1]: 3
29854-> O[0,2]_M[5,2]: 4
29929-> O[0,3]_M[1,0]: 1
29942-> O[0,3]_M[2,2]: 1
29953-> O[0,3]_M[3,2]: 5
29964-> O[0,3]_M[4,2]: 3
29974-> O[0,3]_M[5,1]: 2
29975-> O[0,3]_M[5,2]: 3
30063-> O[0,4]_M[2,2]: 2
30074-> O[0,4]_M[3,2]: 1
30085-> O[0,4]_M[4,2]: 2
30095-> O[0,4]_M[5,1]: 1
30096-> O[0,4]_M[5,2]: 2
```

**Fig. 4** GraphFingerprint of chemical nanocompound $Al_2O_3$ of size 13 Angstroms shown in Fig. 2. The first element of the GraphFingerprint informs about the maximum number of edges (10). The next two elements of the GraphFingerprint inform about the elements are oxygen (atomic number: 8) and aluminium (atomic number: 13). Only the positions of the vector with a non-null value have been selected. Its position appears at the beginning of each line

**Table 6** Mean Square error and its standard deviation achieved in: the data and method in [23], regression on the GraphFingerprint generated by the 3D-structure of data in [23], regressions on the data in [23] concatenated to (represented by symbol '+') the embedding vector generated by GCN [25], generated by Graphlets [8] and generated by Graphormer [9]

| | MSE | STD |
|---|---|---|
| Original experiment in [23] | 0.12 | 0.08 |
| Regression on GraphFingerprint | 0.76 | 0.16 |
| Regression on [23]+GraphFingerprint | 0.08 | 0.08 |
| Regression on [23]+GCN [25] | 0.11 | 0.13 |
| Regression on [23]+Graphlets [8] | 0.34 | 0.23 |
| Regression on [23]+Graphorner [9] | 0.64 | 0.43 |

## 4.2 Toxicity prediction based on global features

Some regression models have been reported for the prediction of toxicity of metal-oxide nanocompounds [22]. Two of them have been selected as a ground truth of our method. The first one is based on a linear regression and in the second one is based on a logistic regression.

**Table 7** Mean and standard deviation of the Accuracy, Precision and Recall achieved in: the data and method in [24], regression on the GraphFingerprint generated by the 3D-structure of data in [24], regressions on the data in [24] concatenated to (represented by symbol '+') the embedding vector generated by GCN [25], generated by Graphlets [8] and generated by Graphormer [9]

| | Accuracy | Precision | Recall |
|---|---|---|---|
| Original experiment in [24] | 0.81 (0.30) | 0.81 (0.25) | 0.98 (0.10) |
| GraphFingerprint | 0.77 (0.40) | 0.77 (0.23) | 0.78 (0.35) |
| [24]+GraphFingerprint | 0.93 (0.43) | 0.92 (0.21) | 0.98 (0.60) |
| [24]+GCN [25] | 0.56 (0.32) | 0.80 (0.15) | 0.88 (0.20) |
| [24]+Graphlets [8] | 0.53 (0.40) | 0.72 (0.22) | 0.89 (0.3) |
| [24]+Graphorner [9] | 0.45 (0.24) | 0.52 (0.17) | 0.75 (0.31) |

The first one [23], presented in 2015, is a linear regression model that predicts the lactate dehydrogenase release (LDH)[4] based on the following five parameters: nanocompound size, nanocompound size in water, nanocompound size in PBS, Concentration and Zeta potential.[5] The model was trained with 33 nanocompounds. 24 of them were $TiO_2$ and 9 of them were ZnO. $TiO_2$ sizes are 30, 45 and 125 nm, and *ZnO* size are 50, 60 and 70 nm. The first row of Table 6 shows the results presented in [23], which we recomputed and checked. In that paper, regression quality metrics were the mean square error and the standard deviation.[6] A *one leave out* training method was used to train the linear regression model.

The second one [24], presented in 2021, is a logistic linear regression model for toxicity assessment of metal-oxide nanoparticles using the following nine physicochemical features: Core nanocompound size, nanocompound size in water, surface charge, surface area, Ec, reaction time, percentage of dose, energy and number of oxygen. The method returns two labels: Toxic and Non-toxic. The model was trained with 484 nanocompounds. 18 of them were $Al_2O_3$ (size 39.7 nm), 18 of them were CuO (size 46.3 nm), 18 of them were $Fe_2O_3$ (size 42.5 nm), 195 of them were $TiO_2$ (sizes 10 nm, 20 nm, 21 nm, 25 nm, 30 nm, 33.7 nm, 35 nm, 40 nm and 125 nm) and 234 of them were ZnO (sizes 7.5 nm, 32 nm, 35.6 nm, 45.3 nm, 60 nm, 86 nm, 100 nm and 115 nm). The first row of Table 7 shows the results presented in [24], which we recomputed and checked. In that paper, quality metrics were Accuracy, Precision and Recall.[7]

[4] LDH is a stable cytoplasmic enzyme that is found in all cells. LDH is rapidly released into the cell culture supernatant when the plasma membrane is damaged, a key feature of cells undergoing apoptosis, necrosis, and other forms of cellular damage.

[5] These parameters and the generated LDH were measured in vitro.

[6] The data extracted from [23] is detailed in the Table 1 in the supplementary material.

[7] The data extracted from [24] is detailed in the Table 2 in the supplementary material.

A *cross-validation partition* with 10 folders training method was used to train the logistic linear regression model.

### 4.3 Toxicity prediction based on GraphFingerprints

We proceeded to analyse the GraphFingerprint representation given the GraphFingerprints generated by the three-dimensional structures of the nanocompounds used in [23] and also in [24]. GraphFingerprints depend only on the type of nanocompound and its size. For this reason, given the 33 different samples used in [23], only six GraphFingerprints were generated. Moreover, given the 484 different samples used in [24], only twenty GraphFingerprints were generated. Figure 5 shows the process of generating a GraphFingerprint given a nanocompound. Circles represent processes and rectangles represent data.

When these GraphFingerprints were generated, we proceeded to learn a linear regression in the case of [23] and a logistic regression in the case of [24]. We finally performed the same type of experiments as depicted in Sect. 4.2. The second row of Tables 6 and 7 show the results obtained given this process.

As it was expected, only the structural information of the nanocompound is not enough to predict its toxicity. This is because several nanocompunds with the same size but different parametres appear in the database and obtain different toxicity levels. For instance, in the case of [23], there are eight $TiO_2$ with size 30nm that have LDH from 0.7 to 1.2. (Table 1 in the supplementary material shows the original data). Another example is ZnO with size 32 nm that appear in [24] (Table 2 in the supplementary material shows the original data). There are 21 examples of it but 6 of them are toxic and the other 15 none.

### 4.4 Toxicity prediction based on global features and GraphFingerprints

In this last experiment we concatenated the original samples in [23] or in [24] with their respective embed vectors generated by GraphFingerprint and also the embedding methods Graph Convolutional Network [25] and Graphlets [8]. That is, each sample in [23] is composed of a vector of five elements. Then, we concatenated these five elements to the corresponding embedded vector. Similarly was done with the data in [24], but in this case, vectors in [24] have ten elements. Finally, we learned the weights of the regression (linear in [23] and logistic in [24]) and we performed the same type of experiments as depicted in Sect. 4.2.

Rows from the third one to the last one of Table 6 and Table 7 show the results obtained given this process. Considering Table 6, we realise that GraphFingerprints concatenated to the original data obtained the lowest MSE and standard deviation. Moreover, note that some of the methods obtain higher or equal MSE, which means that it is not worth using the embedding of the three-dimensional structure in these cases. It is surprising the bad performance of the graph transformer [9]. We believe it is because the positional encoding brings almost no information due to the gaphs have almost constant structure. Considering Table 7, we realise the original data concatenated to GaphFingerprint is the combination that obtains the best accuracy, precision and recall.

We conclude, from these three experiments that:

- Structural information alone is not enough for the toxicity prediction (Balanced accuracy, Precision and Recall are lower in the second row than in the first one in both tables.
- Nevertheless, the structural information represented as a GraphFingerprint helps to increase the quality of the toxicity prediction combined to the global features (The third row returns better validation parameters than first one in both tables).
- Finally, the current embedding methods are not able to capture the structural information of the three-dimensional structure of the nanocompound.

### 4.5 ATENA: a web server to generate GraphFingerprints

The web server ATENA[8] is devoted to analyse metal-oxide nanocompounds from the structural point of view. This means that the three-dimensional information of the compound is required. From an initial description of the nanocompound formated as XYZ file, some toxicity predictions can be performed. Moreover, some analysis can be done, such as the location of specific local patterns inside the nanocompound, which are described in XYZ or GRF format. And also the appearance of combinations of atoms in the shell that are know to be toxic. XYZ and GRF formats are explained and exemplified in the web site.

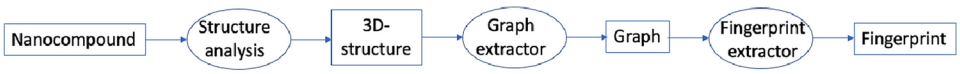


**Fig. 5** The process of generating a GraphFingerprint given a chemical nanocompound

[8] https://atena.urv.cat/model/

The website is composed of three main functions: *NanoFingerprint*, *Toxicity prediction* and *Substructure search*. There is also other material such as examples and general information.

- **NanoFingerprint** This tool embeds in a GraphFingerprint the graph generated by the three-dimensional structure of a nanocompound described in a.XYZ file. The code is public in GitHub.[9] There are some web sites that provide the.XYZ files of some nanocompunds. For instance, *A crystallographic tool for the construction of nanoparticles*[10] reported in [26]. Note in this website, GraphFigerprint is renamed by NanoFingerprint due to the website is specialised in nanocompounds and a new value in Section 1 of GraphFingerprint is incorporated, which is the shell thickness of the nanocompound. It is known that the most external part of the nanocompound, called shell, is the one responsible of some properties, such as the toxicity. This parameter is included as the first element in Section 1 of NanoFingerprint. Thus, this section is composed of six values instead of five. Moreover, since the type of atom *O* is always oxygen, the third value is not the atomic number of oxygen but the size of the nanocompound. The tool returns the GraphFingerprint in a.TXT file format deduced from only the most external atoms of the nanocompound, which are the ones considered to be in its shell (with the slight modification of adding the information of the shell as a first element of the vector and the size of the nanocompound). Note that if an atom has more bonds (represented as edges in the graph) than the maximum number of bonds imposed by the user, this atom is not considered. There is an example of a generated GraphFingerprint by this tool in Fig. 4 (the first element that informs about the shell has been deleted). The imposed shell thickness was 100 (a huge value to assure all the graph is considered) and the maximum number of bonds was 10 (there are not atoms with more bonds in this compound). The.XYZ file represented a $Al_2O_3$ that had a maximum dimension of 13 Angstroms is shown in Fig. 2.
- **Toxicity prediction** This tool computes the toxicity of some nanocompounds with reported models such as the ones explained in Sect. 4.2 and also the models that use GraphFingerprint explained in Sect. 4.4. References and explanations of the models are detailed in the web site.
- **Substructure search** This tool returns the appearance of some specific local structures that could appear in the nanocompound. It is required to introduce the nanocompound in.XYZ format and the local structure in.XYZ or.GRF formats (formats are described in the web site). Moreover, it is also required to introduce the thickness of the shell (if the user wants to search the local structure in the whole compound, a large number can be introduced). The tool uses the sub-graph isomorphism algorithm VF3 [27] and the code was downloaded from Github.[11] As an example, Fig. 6 shows $TiO_2$ compound of size 2 nm and Fig. 7 (right) shows $TiO_2$ compound of 6 Angstrom. Our aim is to search the appearances of the smaller compound in the shell of the larger one. If we impose a thickness shell of 4 Angstrom, Fig. 7 (left) shows the output generated by ATENA. It seems there are four appearances of the smaller compound in the shell of the larger one. Note the output of the server is not the image but a.XYZ file. These images have been generated by the Matlab function *molviewer* although there are other packages for these visualisations.

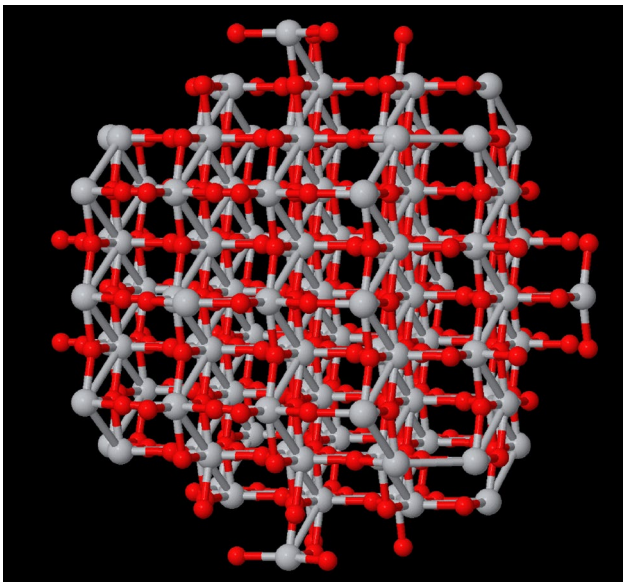

**Fig. 6** Visualisation of $TiO_2$ of 2 nm

## 5 Conclusion

We have presented a new embedding method, called GraphFingerprint, specifically designed to embed graphs that are composed of a large number of almost similar structures and attributes. This embedding has been applied to predict the toxicity of chemical metal-oxide nanocompounds in

[9] https://github.com/ASCLEPIUS-URV/NanoFingerprint

[10] https://nanocrystal.vi-seem.eu/

[11] https://github.com/MiviaLab/vf3lib

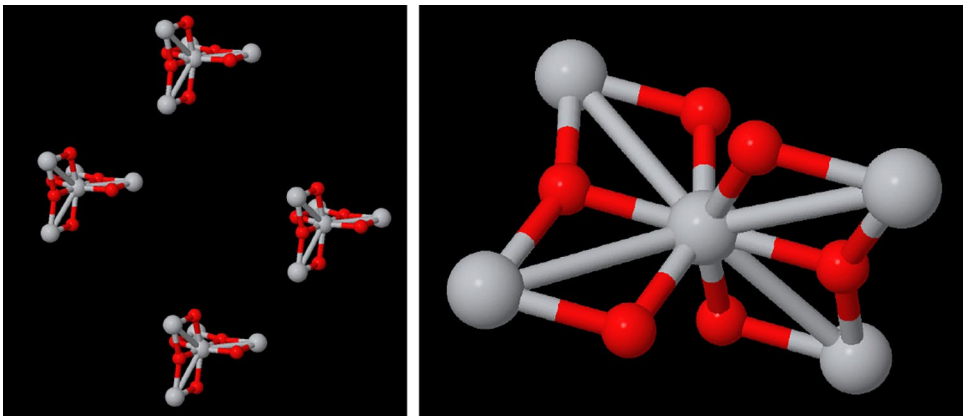

**Fig. 7** Left: Visualisation of the four appearances of $TiO_2$ of 06A (right) in the shell (thickness 4Angstrom) of $TiO_2$ of 2 nm (Fig. 6)

a regression scenario and also a classification scenario. Our proposal has been tested with different databases presented in other papers and it has achieved the highest accuracy when combined with global features of the nanocompound. We have seen that the other embedding methods are not able to properly represent the attributed graphs that represent the three-dimensional structure of the nanocompounds. Note we have also realised that the three-dimensional structure, without global features, has not information enough to properly classify the nanocompound or deduce its regression.

Thus, we conclude that, on the one hand, our embedding method applied to classification and regression of nanocompounds achieves the highest accuracy compared to other embedding methods, such as graph convolutional networks, graph autoencoders or older and classical embedding techniques. On the other hand, the three-dimensional structure of the compounds seems not to have information enough to properly deduce the toxicity and, for this reason, the best accuracy appears when the global information is concatenated to the GraphFingerprint.

## 6 Future work

We are currently using GraphFingerprints to analyse social networks. Social networks are easily represented as graphs and some of them have our required features. For instance, nodes have only two classes (the two most voted parties in some countries) and usually the local structures (people friendship) are almost constant. We realised the current graph embedding methods do not extract the main information of these graphs, as it happened with the three-dimensional structure of nanocompounds.

Moreover, we are applying graph generative techniques, specifically Graph Adversarial Networks, GANs, to automatically generate GraphFingerprints. The aim is to increase a learning and testing database of GraphFingerprints since the generation of graphs from which to extract GraphFingerprints could be hard in some applications.

Since these are research topics in process, our findings will be presented in another paper.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s10044-024-01366-w.

**Funding** European projects NanoInformatix: H2020-NMBP-14-2018-814426 and SbD4Nano: H2020-NMBP-TO-IND-2019-862195; AGAUR research group 2021SGR-00111 "ASCLEPIUS: Smart Technology for Smart Healthcare"; Spanish project NextPandemics: PID2022-138327OB-I00.
Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

**Availability of data** Publicly available data is used presented in [23, 24].

**Code Availability** Code available at https://github.com/ASCLEPIUS-URV/NanoFingerprint

## Declarations

**Conflict of interest** No conflict of interest.



## References

1. Foggia P, Percannella G, Vento M (2014) Graph matching and learning in pattern recognition in the last 10 years. Int J Pattern Recogn Artif Intell. https://doi.org/10.1142/S0218001414500013

2. Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(3):265–298. https://doi.org/10.1142/S0218001404003228
3. Garcia-Hernandez C, Fernández A, Serratosa F (2019) Ligand-based virtual screening using graph edit distance as molecular similarity measure. J Chem Inf Model 59(4):1410–1421
4. Serratosa F, Cortés X (2015) Graph edit distance: Moving from global to local structure to solve the graph-matching problem. Pattern Recogn Lett 65:204–210
5. Serratosa F (2021) Redefining the graph edit distance. SN Comput Sci. https://doi.org/10.1007/s42979-021-00792-5
6. Reiser P, Neubert M, Eberhard A, Torresi L, Zhou C, Shao C, Metni H, Hoesel C, Schopmans H, Sommer T, Friederich P (2022) Graph neural networks for materials science and chemistry. Commun Mater. https://doi.org/10.1038/s43246-022-00315-6
7. Fadlallah S, Julià C, Serratosa F (2022) Graph regression based on graph autoencoders. In: Krzyzak A, Suen CY, Torsello A, Nobile N (eds) Structural, syntactic, and statistical pattern recognition. Springer, Cham, pp 142–151
8. Dutta A, Riba P, Llados J, Fornes A (2020) Hierarchical stochastic graphlet embedding for graph-based pattern recognition. Neural Comput Appl 32(15):11579–11596. https://doi.org/10.1007/s00521-019-04642-7
9. Ying C, Cai T, Luo S, Shuxin Z, Ke G, He D, Shen Y, Liu T-Y (2021) Do transformers really perform badly for graph representation?
10. Serratosa F (2020) A general model to define the substitution, insertion and deletion graph edit costs based on an embedded space. Pattern Recognit Lett 138:115–122. https://doi.org/10.1016/j.patrec.2020.07.010
11. Serratosa F (2014) Speeding up fast bipartite graph matching through a new cost matrix. Int J Pattern Recogn Artif Intell 29:1550010. https://doi.org/10.1142/S021800141550010X
12. Serratosa F (2014) Fast computation of bipartite graph matching. Pattern Recogn Lett 45:244–250
13. Gibert J, Valveny E, Bunke H (2012) Graph embedding in vector spaces by node attribute statistics. Pattern Recogn 45(9):3072–3083
14. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS (2021) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24. https://doi.org/10.1109/TNNLS.2020.2978386
15. Kipf TN (2020) Deep learning with graph-structured representations. PhD thesis, University of Amsterdam
16. Wang J, Liang J, Yao K, Liang J, Wang D (2022) Graph convolutional autoencoders with co-learning of graph structure and node attributes. Pattern Recogn 121:108215. https://doi.org/10.1016/j.patcog.2021.108215
17. Lin M, Wen K, Zhu X, Zhao H, Sun X (2023) Graph autoencoder with preserving node attribute similarity. Entropy. https://doi.org/10.3390/e25040567
18. Çetin YA, Martorell B, Serratosa F, Aguilera-Porta N, Calatayud M (2022) Analyzing the tio2 surface reactivity based on oxygen vacancies computed by dft and dftb methods. J Phys Condens Matter 34(31):314004
19. Rica E, Álvarez S, Serratosa F (2021) Ligand-based virtual screening based on the graph edit distance. Int J Mol Sci 22(23):12751
20. Garcia-Hernandez C, Fernández A, Serratosa F (2020) Learning the edit costs of graph edit distance applied to ligand-based virtual screening. Curr Top Med Chem 20(18):1582–1592
21. Garcia-Hernandez C, Fernández A, Serratosa F (2019) Ligand-based virtual screening using graph edit distance as molecular similarity measure. J Chem Inf Model 59(4):1410–1421. https://doi.org/10.1021/acs.jcim.8b00820
22. Lamon L, Asturiol D, Vilchez A, Ruperez-Illescas R, Cabellos J, Richarz A, Worth A (2019) Computational models for the assessment of manufactured nanomaterials: Development of model reporting standards and mapping of the model landscape. Comput Toxicol 9:143–151. https://doi.org/10.1016/j.comtox.2018.12.002
23. Papa E, Doucet JP, Doucet-Panaye A (2015) Linear and non-linear modelling of the cytotoxicity of tio2 and zno nanoparticles by empirical descriptors. SAR QSAR Environ Res 26(7–9):647–665. https://doi.org/10.1080/1062936X.2015.1080186
24. Subramanian NA, Palaniappan A (2021) Nanotox: development of a parsimonious in silico model for toxicity assessment of metal-oxide nanoparticles using physicochemical features. ACS Omega 6(17):11729–11739. https://doi.org/10.1021/acsomega.1c01076
25. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th International conference on learning representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl
26. Chatzigoulas A, Karathanou K, Dellis D, Cournia Z (2018) Nanocrystal: a web-based crystallographic tool for the construction of nanoparticles based on their crystal habit. J Chem Inf Model 58(12):2380–2386. https://doi.org/10.1021/acs.jcim.8b00269
27. Carletti V, Foggia P, Greco A, Vento M, Vigilante V (2019) Vf3-light: a lightweight subgraph isomorphism algorithm and its experimental evaluation. Pattern Recogn Lett 125:591–596. https://doi.org/10.1016/j.patrec.2019.07.001