Treelet kernel incorporating cyclic, stereo and inter pattern information in Chemoinformatics

Benoit Gaüzère, Pierre-Anthony Grenier, Luc Brun, Didier Villemin

Treelet kernel incorporating cyclic, stereo and inter pattern information in Chemoinformatics

Benoit Gaüzère [1], Pierre-Anthony Grenier[1], Luc Brun[1], and Didier Villemin[2]

[1]GREYC UMR CNRS 6072, Caen, France
[2]LCMT UMR CNRS 6507, Caen, France
firstname.lastname@ensicaen.fr

August 2014

Abstract

Chemoinformatics is a research field concerned with the study of physical or biological molecular properties through computer science's research fields such as machine learning and graph theory. From this point of view, graph kernels provide a nice framework which allows to naturally combine machine learning and graph theory techniques. Graph kernels based on bags of patterns have proven their efficiency on several problems both in terms of accuracy and computational time. Treelet kernel is a graph kernel based on a bag of small subtrees. We propose in this paper several extensions of this kernel devoted to chemoinformatics problems. These extensions aim to weight each pattern according to its influence, to include the comparison of non-isomorphic patterns, to include stereo information and finally to explicitly encode cyclic information into kernel computation.

1 Introduction

Chemoinformatics is a science field lying at the edge of chemical and computer science which aims to study, analyze, store and predict chemical data through informational techniques. Two problematics addressed by chemoinformatics correspond to Quantitative Structure-Activity Relationship (QSAR) and Quantitative Structure-Property Relationship (QSPR) problems. QSAR and QSPR are based on the similarity principle [1] which states that two structurally similar molecules should have similar activities and properties.

A molecule is usually encoded by its molecular graph. A molecular graph is defined as a labeled graph $G = (V, E, \mu, \nu)$, where the unlabeled graph (V, E) encodes the structure of the molecule while μ maps each vertex to an atom's label corresponding to atom's chemical element. Labeling function ν characterizes each edge by a type of bond between two atoms (single, double, triple or aromatic). Considering this molecular representation and the similarity principle, QSAR/QSPR problems rely on defining a similarity measure between molecular graphs.

A first family of methods introduced within QSAR/QSPR field is based on the correlation between a set of molecular descriptors and a given molecular property (e.g. molecule's boiling point). Descriptor sets are encoded by vectors and may be computed from structural information [2], physical properties or biological activities. These vectors may be used within any statistical machine learning algorithm in order to predict molecule's properties. Such a scheme allows to benefit from the large set of tools available within the statistical machine learning framework. However, the definition of a fixed size vector from a molecule, i.e. a molecular graph, induces a loss of information. Moreover, for each application, the definition of a vectorial description of each molecule is based on chemical expert's heuristics. A slightly different approach is based on graph embedding. Within this framework, a vectorial description of a graph is automatically built from its encoding using, for example, the spectral analysis of graphs [3]. In this last case, the embedding is deduced from the analysis of the eigenvectors and eigenvalues of the adjacency matrix.

A second family of methods, based on graph theory [4], may be decomposed in two subfamilies. The first subfamily [5], related to the data mining field, aims to discover subgraphs having a large difference of frequencies between sets of positive and negative examples. We can note that this first subfamily is mainly restricted to classification problems. The second subfamily [6], bases molecular graph similarity measures on graph edit distance. However, since graph edit distance does not define an euclidean distance [7], similarity measures based on graph edit distance do not correspond to a direct explicit nor implicit embedding. An embedding may nevertheless be obtained by regularizing the edit distance or by using distances to a set of prototype graphs [6]. Graph edit distance can also be directly combined with a restricted set of machine learning methods such as k-nearest neighbors or k-medians algorithms.

Graph kernels can be understood as symmetric graph similarity measures. Using a positive definite kernel k, the value $k(G, G')$, where G and G' encode two graphs, corresponds to a scalar product between two vectors $\psi(G)$ and $\psi(G')$ in a Hilbert space. Therefore, such a similarity measure may be used in conjunction with machine learning methods which may access to input data only through scalar products (such as SVM). Graph kernel framework provides thus a natural connection between structural pattern recognition and graph theory on one hand and statistical pattern recognition on the other hand.

A large family of graph kernels used in chemoinformatics is based on the definition of a bag of patterns for each graph and deduces graph similarity from similarity between bags. Some bags of patterns are defined by linear patterns. For example, marginalized kernel [8] defines a graph kernel based on a comparison between sets of random walks extracted from each graph. Tree pattern kernel [9] defines a graph kernel using a set of tree patterns instead of walks. This last kernel allows to encode more structural information than kernels based on linear patterns such as random walks.

A common drawback of kernels based on random walks and tree patterns is that the global similarity between two graphs is based on an implicit enumeration of their common patterns. An implicit enumeration does not allow to analyze the relevance of each pattern according to a given dataset and a particular property. Weisfeiler-Lehman kernel [10] corresponds to a kernel based on a subset of tree patterns, computable in linear time, which provides an explicit distribution of each pattern within a graph. However, authors do not propose a method to weight each pattern. Treelet kernel [11] defines a graph kernel based on a explicit set of patterns, called treelets, defined as all labeled subtrees having at most 6 nodes. This kernel can be used with a treelet selection step which aims to select relevant treelets according to a given property.

Kernels based on bags of patterns compute a similarity between two molecular graphs by comparing the number of occurrences of a given set of patterns in each graph. This approach only compares strictly isomorphic patterns. However, similarity principle established between molecular graphs can be transposed to patterns, i.e. similar patterns may have a similar influence on molecular properties. Considering this hypothesis, the shingled Weisfeiler-Lehman subtree kernel [12] consists in including the comparison of similar tree patterns into kernel computation. However, this kernel is limited to the comparison of structurally isomorphic patterns and the set of patterns is restricted to balanced trees.

However, kernels based on linear or tree patterns do not encode any cyclic information. Molecular cycles correspond to an important molecular characteristic since they reduce the atom's degrees of freedom and hence influence molecular properties. Based on this last point, cyclic systems are used by chemical experts in order to classify molecules into several distinct molecular families, each family having similar chemical and biological properties. Cyclic pattern kernel [13] aims to encode cyclic similarity between molecular graphs. This kernel is defined as a sum of two subkernels, a first one encoding acyclic similarity and a second one encoding cyclic similarity. However, the cyclic information captured by this kernel is restricted to the number of common cycles of two graphs and hence does not encode adjacency relationships between molecular cycles.

Finally, most of existing graph kernels do not take into account stereoisomerism. Stereoisomerism is a molecular property which distinguishes two molecules, called stereoisomers, encoded by a same molecular graph but which differs by the relative positioning of their atoms. Stereoisomers can have different properties which can not be accurately predicted by usual graph kernels which consider these molecules as identical. The adaptation of the tree pattern kernel [14] in order to encode stereo information constitutes a noticeable exception. However, this method does not provide a clear separation between patterns encoding stereoisomerism and patterns which do not.

This article is an extension of two previous papers [15, 16] which extend treelet kernel by explicitly taking into account cyclic and chiral information of molecules. In this article, we propose a unified presentation of treelet kernel including a detailed description of the extensions proposed in [15, 16]. In addition, we propose an original extension of treelet kernel which includes the contribution of similar patterns. Moreover, we propose to explain in detail a pattern weighting scheme based on multiple kernel learning and more complete experiments. Our paper is organized as follows: First, the treelet kernel construction scheme is described in section 2 together with a weighting scheme based on multiple kernel learning [17] which allows to compute an optimal weight for each pattern according to a given property. Second, section 3 aims to define a kernel which includes the comparison of similar patterns. Then, section 4 aims to include stereo information into graph kernel. Finally, we propose to encode cyclic information in two different ways. The first one consists in encoding the cyclic system of a molecule by its relevant cycle graph (section 5.2) which allows to encode relationships between cycles of a molecule. The second representation, called relevant cycle hypergraph (section 5.3), allows to encode relationships between cyclic and acyclic parts into an unique molecular representation. Treelet kernel has been adapted to the comparison of these two new molecular representations. Finally, all proposed extensions are evaluated in section 6 on several chemical data sets.

2 Treelet kernel

The treelet kernel [11] is a graph kernel defined as a convolution kernel [18] between bags of patterns extracted from two graphs. The set of extracted patterns, called treelets, is composed of all labeled trees with a number of nodes lower than or equals to 6. The set of treelet's structures, i.e. the set of all unlabeled trees associated to treelets, is displayed in figure 1. This set of patterns allows us to define an ad hoc enumeration algorithm which encodes most of the relevant structural information encoded in a molecular graph. Indeed, graph fragment approach experiments [19] have observed that no significant gain is obtained with structures having more than 6 nodes. In order to compute treelet kernel, we have to enumerate the set of treelets of each graph. The first step of this enumeration consists in enumerating the set of treelet's structures (figure 1) of a graph. This structure identification step consists in two sub steps: First, linear patterns are enumerated by a depth first traversal performed from each node. In order to enumerate all paths having at most 6 nodes, depth first traversal is stopped after encountering 5 edges. Then, non linear treelets are enumerated using a neighborhood analysis of n-star nodes which are defined as nodes having

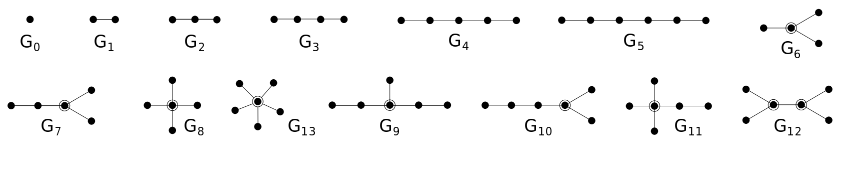


Figure 1: Set of treelet's structures

a degree equals to n (G_6, G_8 and G_{13} treelet's structures in figure 1). Considering these first structures, others treelet's structures are enumerated using the neighborhood of nodes adjacent to n-stars. Therefore, structures G_7, G_9, G_{10} and G_{12} are enumerated from G_6. Similarly, G_{11} is retrieved from G_8. This structure identification step allows to associate an index to each enumerated subgraph. This index identifies the structure of each treelet in a canonical way.

Once this first step is performed, a key encoding the labels of each treelet is computed in order to distinguish treelets having a same structure index but different labels. Considering linear treelets, this key is defined as a sequence of labels encountered during a traversal of the associated path. Since there is two possible path traversals, this sequence is defined as the lowest one according to lexicographic order. For each non linear treelet structure, our key's construction scheme requires to root each treelet. To this end we use the Morgan numbering [20] which maps each node to an integer depending on node's degrees. Considering this numbering, each tree is rooted on the node having the highest Morgan number. Then each set of child nodes is ordered using a subkey defined from its Morgan numbering and the concatenation of the child's label, the label of the edge connecting it to its father and the key of the subtree rooted on this child. Given this ordered rooted tree, the key is defined as a sequence of labels encountered during a depth first traversal. This traversal provides thus a sequence of nodes and edges labels which is unique for two isomorphic treelets. Conversely, we have shown [11] that two treelets with a same index (i.e. a same structure) and a same key are isomorphic. Therefore, the concatenation of treelet's structure index and treelet's key defines a unique code for each treelet which allows to perform an explicit enumeration of all treelets included within a graph. Based on this enumeration, we define a function $f_t : \mathcal{G} \to \mathbb{R}_+$ which encodes the number of occurrences of treelet $t \in \mathcal{T}$ in a graph:

$$\forall\ t \in \mathcal{T}, f_t(G) = |(t \trianglelefteq G)| \tag{1}$$

where $\mathcal{T}$ denotes the set of treelets and $\trianglelefteq$ the subgraph isomorphism relationship. Then, similarity between treelet distributions is computed using a sum of subkernels between treelet's frequencies:

$$k_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k(f_t(G), f_t(G')) \overset{not.}{=} \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k_t(G, G') \tag{2}$$

where $\mathcal{T}(G)$ encodes the set of treelets extracted from G, $k_t(G, G') = k(f_t(G), f_t(G'))$ and $k(.,.)$ corresponds to any positive definite kernel between real numbers such as linear kernel, gaussian kernel or intersection kernel. Each subkernel k_t encodes thus a similarity between frequencies of a given treelet t in each graph. Note that, unlike tree pattern or random walks kernels, this kernel explicitly enumerates subtrees by computing the number of occurrences of each pattern. This explicit enumeration allows us to apply a treelet weighting step, as defined in section 2.1.

2.1 Pattern weighting

Chemical experts aim to identify particular subgraphs which may be responsible for a particular molecular activity. The presence or absence of one of these particular subgraphs, called pharmacophores, is an important characteristic to determine if a molecule is toxic or not. Considering graph kernels based on bags on patterns, each pattern may be seen as a potential pharmacophore. It may thus be interesting to weight each pattern according to its influence on a given property. A relevant pattern may thus be associated to an important weight whereas patterns which do not encode any information may be removed from kernel computation.

Considering a kernel based on a bag of patterns $\mathcal{P}$ and defined as a sum of subkernels, each subkernel encodes graph similarity according to a given pattern $p \in \mathcal{P}$. Pattern weighting relies on computing a weight for each term of the sum. A weighted kernel is defined as:

$$k_{\text{weight}}(x, x') = \sum_{p \in \mathcal{P}} d(p) * k_p(x, x') \tag{3}$$

where $d : \mathcal{P} \rightarrow \mathbb{R}_+$ encodes the influence of each pattern into kernel computation. An high $d(p)$ encodes an high influence and corresponds to an high contribution of pattern p. Conversely, a weight $d(p)$ close to or equal to 0 relies to remove the contribution of pattern p from kernel computation.

Tree pattern kernel [9] includes a pattern weighting function. This weighting is an *a priori* parameter which allows to favor structurally complex tree patterns or linear patterns depending on parameter value. Treelet kernel can be combined with iterative pattern selection method [11] which allows to select or not a given pattern into kernel computation. Conversely to *a priori* pattern weighting method proposed by tree pattern kernel [9], this last method computes a weighting function according to a particular property. However, this pattern selection step is only binary, computationally costly and non optimal.

2.1.1 Multiple kernel learning

Considering a pattern weighting problem as defined by equation 3, Multiple Kernel Learning (MKL) methods aim to compute an optimal weighting function d according to a given dataset and a property to predict. Considering a finite set of M patterns, weighting function d is encoded by a vector $\vec{d} \in \mathbb{R}_+^M$ where each coordinate d_m encodes a weight associated to a pattern. The weighted kernel is then defined as:

$$k_{\text{MKL}}(x, x') = \sum_{m=1}^{M} d_m * k_m(x, x') \tag{4}$$

where k_m corresponds to a subkernel encoding a graph similarity measure according to a particular pattern. Multiple kernel learning methods consist in computing an optimal vector d according to a prediction task. Simple MKL [21] consists in optimizing a SVM problem by computing an optimal weighting for each sub kernel. Considering a kernel as defined in equation 4, Simple MKL consists in solving:

$$\underset{d}{\text{minimize}}\ J(d) \text{ such that } \begin{cases} d_m > 0,\ \forall i \in \{1, \dots, M\} \\ \sum_{m=1}^{M} d_m = 1 \end{cases} \tag{5}$$

with

$$J(d) = \begin{cases} \min\limits_{w,b,\xi} & \frac{1}{2}\sum_{m=1}^{M} \frac{1}{d_m}\|w_m\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to:} & y_i(\sum_{m=1}^{M}\langle w_m, \Phi_m(x)\rangle + b) \geq 1 - \xi_i, \forall i \in \{1, \dots, n\} \\ & \xi_i \geq 0,\ \forall i \in \{1, \dots, n\} \end{cases} \tag{6}$$

where $\Phi_m(x)$ corresponds to the embedding function associated to k_m. This minimization problem is the same type of problem as solved by a standard SVM by optimizing vector d up to constraints defined by equation 5. Minimization problem defined in equation 5 includes a first constraint on vector d which is defined as a positivity constraint. This constraint ensures that the weighted kernel (equation 4) is positive definite. The second constraint imposes to the L_1 norm of vector d to be equals to 1. This constraint induces a sparse vector d and thus allows to keep only the most relevant patterns into the weighted kernel while removing irrelevant ones.

Simple MKL [21] problem is resolved by alternating a classical SVM resolution together with a projected gradient descent according to direction $\frac{\partial J}{\partial d_m}$ for each element of vector d. Using a projected gradient method allows us to minimize the objective function while ensuring that constraints of sparsity and positivity defined on vector d are fulfilled. Generalized MKL [22] proposes a multiple kernel learning algorithm based on the same principle but where the objective function is penalized by $\sigma\|d\|_1$ instead of an equality constraint on $\|d\|_1$. This regularization allows to tune the degree of sparsity of d.

In order to deal with thousands of patterns, one have to either compute each gram matrix for each sub kernel at each iteration or to store each gram matrix. When considering thousands of patterns and graphs, these two options induce too much computational time or memory space to be applicable. In order to handle such datasets, infinite MKL [23] defines a MKL method based on Simple MKL which only considers a sub set of sub kernels at each iteration. This set of active kernels is updated between two Simple MKL iterations until convergence or a maximal numbers of active kernels is reached. This algorithm allows to consider an high, possibly infinite, number of subkernels and to select most relevant ones according to a particular property to predict.

Initially, multiple kernel learning methods have been defined on classification problems. However, they can easily be adapted to SVM for regression. This adaptation mainly consists in replacing SVM objective function by SVM for regression objective function. Obviously, gradient values must be updated accordingly. Therefore, multiple kernel learning methods allow to compute a weighting for a set of kernels for either a classification or a regression problem. This weight is optimal according to the used training set and not defined *a priori* or using only structural information. When applied together with treelet kernel, each weight is associated to a given treelet. On one hand, this weighting may allow to increase prediction accuracy obtained by treelet kernel. On the other hand, computed weights may be seen as a measure of the influence of each treelet according to the predicted property. This measure of influence may be used by chemical experts in order to understand chemical properties. Indeed, a treelet associated to an high weight may be considered as a potential pharmacophore.

3 Including cross pattern information

As mentioned in section 1, if we consider the hypothesis that similar molecules have similar properties, we should consider that similar substructures may have a similar influence on molecular properties. Recently, Shervashidze has proposed to adapt Weisfeiler-Lehman kernel in order to include comparison of non isomorphic subtree patterns [12]. This comparison is based on Jaccard coefficient which defines a similarity measure between sets. Considering this similarity measure, Shervashidze has proposed an efficient computation in order to compare nodes' neighbourhoods. However, this approach only compares structurally isomorphic patterns and the set of patterns is restricted to balanced trees.

In order to include non-isomorphic treelet comparison into treelet kernel, let us recall convolution kernel theory. Haussler's convolution kernels [18] are defined on objects $x \in \mathcal{X}$ which can be associated to a decomposition into finite sets $\mathcal{X}_x$. Considering a subkernel $k_s : \mathcal{X}_x \times \mathcal{X}_x \rightarrow \mathbb{R}$, Haussler's convolution kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is defined as follows:

$$K(x,y) = \sum_{(x',y') \in \mathcal{X}_x \times \mathcal{X}_y} k_s(x',y') \tag{7}$$

By considering a decomposition $\mathcal{X}_G = \{(t, f_t(G)) \mid t \trianglelefteq G\}$ of each graph and a tensor product $(k \otimes k')$ of two kernels $k' : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$ and $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, treelet kernel $k_{\mathcal{T}}$ (equation 2) can be reformulated as a convolution kernel k_{inter}:

$$\begin{aligned} k_{\text{inter}}(G,G') &= \sum_{\substack{(t,f_t(G)) \in \mathcal{X}_G \\ (t',f_{t'}(G')) \in \mathcal{X}_{G'}}} (k' \otimes k)(t, f_t(G), t', f_{t'}(G')) \\ k_{\text{inter}}(G,G') &= \sum_{\substack{(t,f_t(G)) \in \mathcal{X}_G \\ (t',f_{t'}(G')) \in \mathcal{X}_{G'}}} k'(t,t') k(f_t(G), f_{t'}(G')) \end{aligned} \tag{8}$$

In treelet kernel definition (equation 2), k' is defined such that comparisons are restricted to isomorphic treelets, i.e. $k'(t,t') = 1 \Leftrightarrow t \simeq t'$, and 0 otherwise. In this section, we propose to define a kernel k' between treelets which relaxes this restriction and includes the comparison of similar patterns.

3.1 Inter treelet kernel based on edit distance

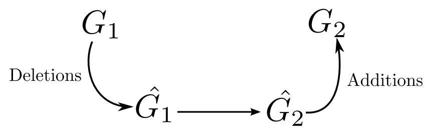
Considering the hypothesis that similar patterns may have a similar influence, kernel k' between treelets must encode a similarity measure between t and t'. This similarity measure may be defined from graph edit distance [24]. Graph edit distance is defined as the sequence of edit operations transforming G into G' with a minimal cost. A sequence of edit operations, called edit path, may include vertex or edge addition, removal and relabeling. Given a cost function $c(.)$ associated to each operation, the cost of a sequence of edit operations is defined as the sum of each elementary operation's costs. A high edit distance corresponds to a low similarity between two graphs while a small one encodes a strong similarity. According to [24], the computational cost of the exact edit distance grows exponentially with the size of graphs. To overcome this problem, Fankhauser et al. [25] propose a method to compute an approximate edit distance in $O(n^3)$ where n is equal to the number of nodes of both graphs. Such an edit distance computation provides an efficient way to compute an approximate edit distance between graphs at the cost of a lower precision.

3.1.1 Exact treelet edit distance

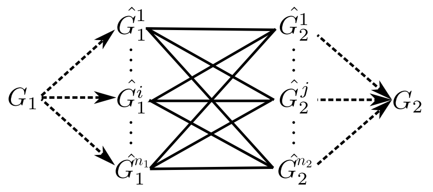
Exact edit distance is hard to compute when considering the whole set of possible graphs. Given a finite set of n structures $B = \{(V_1, E_1), \ldots, (V_n, E_n)\}$, we restrict our study to sets of graphs D such that for any graph $G = (V, E, \mu, \nu) \in D$ we have $(V, E) \in B$. We show in the remaining of this section that within this framework, exact edit distance may be computed within a reasonable computational time using ad hoc methods.

In order to present such methods, let us introduce some definitions. A graph $G' = (V', E', \mu', \nu')$ is a structural subgraph of $G = (V, E, \mu, \nu)$, denoted $G' \sqsubseteq_s G$, iff $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. In addition, if $\mu'_{|V'} = \mu$ and $\nu'_{|E'} = \nu$, $g_|$ denoting the restriction of function g to a particular domain, then G' is a subgraph of G, denoted $G' \sqsubseteq G$. A graph $G = (V, E, \mu, \nu)$ is structurally isomorphic to a graph $G' = (V', E', \mu', \nu')$, denoted $G \simeq_s G'$ iff it exists a bijective function $f : V \to V'$ such that $(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$. If $\mu' \circ f = \mu$ and $\nu' \circ f = \nu$, then G is isomorphic to G', denoted $G \simeq G'$. If $G = G'$ then f is called an automorphism. If f is only injective then it exists a subgraph isomorphism between G and G'. A graph $\hat{G}$ is a maximal common subgraph of G_1 and G_2 if it is a subgraph of G_1 and G_2 and if it is not a sub graph of any other common subgraph of G_1 and G_2. A graph $\hat{G}$ is called a maximum common subgraph of G_1 and G_2 if it is a common subgraph of G_1 and G_2 with a maximal number of nodes. The notions of maximal structural subgraph and maximum structural subgraph are defined in the same way but without conditions on the mapping of labels of both graphs.

Under mild assumptions [26], the sequence of edit operations encoding an edit path can be ordered into a sequence of deletions, substitutions and additions as illustrated in figure 2(a). The first sequence transforms the initial graph G_1 into one of its subgraphs $\hat{G}_1$ by deleting a set of nodes corresponding to $V_1 \setminus \hat{V}_1$



(a) Different steps describing an edit path.



(b) Edit paths passing through maximum common structural subgraphs $\{\hat{G}_1^1, \ldots, \hat{G}_1^{n_1}\}$ and $\{\hat{G}_2^1, \ldots, \hat{G}_2^{n_2}\}$. Dashed lines correspond to structural operations, other to substitutions.

Figure 2: Edit path scheme(a) and edit paths passing through maximum common structural subgraphs(b).

and a set of edges corresponding to $E_1 \setminus \hat{E}_1$. The second sequence encodes a set of substitution operations which allows to transform $\hat{G}_1$ into $\hat{G}_2$. This set of substitutions corresponds to a one to one matching between $\hat{V}_1$ and $\hat{V}_2$ on one hand and between $\hat{E}_1$ and $\hat{E}_2$ on the other hand. Substitutions matching two elements having a same label correspond to identical substitutions and are associated to a null cost since they do not induce any changes. Finally, the last sequence of operations corresponds to a set of addition operations required to obtain G_2 from $\hat{G}_2$. Note that the set of operations transforming $\hat{G}_1$ into $\hat{G}_2$ is only composed of substitutions which do not alter graph structures. Therefore, $\hat{G}_1$ and $\hat{G}_2$ have a same structure and correspond to two structurally isomorphic subgraphs of G_1 and G_2.

Edit operation costs are defined as non-negative constant functions for edges (c_{e*}) and nodes (c_{v*}) deletions (c_{*d}), insertions (c_{*i}) or substitutions (c_{*s}). Using the different steps depicted in figure 2(a) and cost functions previously defined, the cost associated to an edit path is equal to:

$$\gamma(P) = |V_1 - \hat{V}_1| c_{vd} + |E_1 - \hat{E}_1| c_{ed} + V_f c_{vs} + E_f c_{es} + |V_2 - \hat{V}_2| c_{vi} + |E_2 - \hat{E}_2| c_{ei} \quad (9)$$

where V_f, resp. E_f, corresponds to the number of non identical substitutions on nodes, resp. edges, required to transform $\hat{G}_1$ into $\hat{G}_2$. Bunke has shown that under some slightly different conditions on edge operations, constraining the costs to $c_{vd} + c_{vi} < c_{vs}$ and $c_{es} < c_{vs}$ induces that $\hat{G}_1 \simeq \hat{G}_2$ correspond to a maximum common subgraph of G_1 and G_2 [27]. However, maximum common subgraph of two graphs depends both on structural and labeling information which does not allow us to use our assumption that the set of structures encoding graphs is finite and known *a priori*. In this section, we propose to study different assumptions on edit costs leading to an efficient algorithm to compute an exact edit distance.

Proposition 1 *Given two graphs G_1 and G_2, let us denote by δ_v the number of nodes of their maximum structural common subgraph and by δ_e the maximal number of edges of their structural common subgraphs. If $\frac{c_{vd}+c_{vi}}{c_{vs}} \geq \delta_v + \frac{c_{es}}{c_{vs}}\delta_e$ and $\frac{c_{ed}+c_{ei}}{c_{es}} \geq \delta_e + \frac{c_{vs}}{c_{es}}\delta_v$, then $\hat{G}_1$ is a maximal common structural subgraph of G_1 and G_2.*

Proof 1 *Proof of proposition 1 can be found in [26].*

Considering two graphs G_1 and G_2, this first proposition ensures that an optimal edit path transforming G_1 into G_2 passes through one of their maximal common structural subgraphs. Since the maximal common structural subgraph does not depend on labeling, the set of maximal common structural subgraphs may be pre computed between any pair of structures belonging to B. Nevertheless, this number of maximal common structural subgraphs may remain large hence forbidding an efficient pre computation of the exact edit distance. However, by further restricting cost conditions, we obtain a relationship involving a reduced set of substructures:

Proposition 2 *Let us suppose that $c_{ed} = c_{ei} = 0$ and $c_{es} \leq c_{vs}$. Given two graphs G_1 and G_2, let us further denote by δ_v the number of vertices of their maximum common structural subgraphs and by δ_e the maximal number of edges of all maximum common structural subgraphs. Then if $\frac{c_{vd}+c_{vi}}{c_{vs}} \geq \delta_v + \delta_e$, $\hat{G}_1$ is a maximum common structural subgraph of G_1 and G_2.*

Proof 2 *Proof of proposition 2 can be found in [26].*

Proposition 2 states that under some hypothesis on the costs c_{*d}, c_{*i} and c_{*s}, any optimal edit path between two graphs G_1 and G_2 should pass through one of their maximum common structural subgraphs. Let us consider two graphs G_1 and G_2 sharing only one maximum common structural subgraph $\hat{G} = (\hat{V}, \hat{E})$. Let us denote as $\{\hat{G_1^0}, \dots, \hat{G_1^i}, \dots, \hat{G_1^{n_1}}\}$ and $\{\hat{G_2^0}, \dots, \hat{G_2^i}, \dots, \hat{G_2^{n_2}}\}$ the sets of subgraphs of G_1 and G_2 structurally isomorphic to $\hat{G}$ (figure 2(b)). Considering proposition 2, any optimal edit path P between G_1 and G_2 should pass through one $\hat{G_1^i}$ and $\hat{G_2^j}$. The cost associated to P can be decomposed into two parts: a structural cost $\gamma_{struc}(P)$, corresponding to insertion and deletion operations, and a substitution cost $\gamma_{label}(P)$ corresponding to label substitutions required to transform $\hat{G_1^i}$ into $\hat{G_2^j}$:

$$\gamma(P) = \gamma_{struc}(P) + \gamma_{label}(P) \tag{10}$$

Following equation 9, we have:

$$\begin{cases} \gamma_{struc}(P) = |V_1 - \hat{V}_1| c_{vd} + |E_1 - \hat{E}_1| c_{ed} + |V_2 - \hat{V}_2| c_{vi} + |E_2 - \hat{E}_2| c_{ei} \\ \gamma_{label}(P) = V_f c_{vs} + E_f c_{es} \end{cases} \tag{11}$$

For any $i \in \{1, \ldots, n_1\}$, since $\hat{G_1^i} \sqsubseteq G_1$, we have $\hat{V_1^i} \subseteq V_1$ and $\hat{E_1^i} \subseteq E_1$. Therefore:

$$\begin{cases} |\hat{V_1^i} - V_1| &= |V_1| - |\hat{V_1^i}| &= |V_1| - |\hat{V}| \\ |\hat{E_1^i} - E_1| &= |E_1| - |\hat{E_1^i}| &= |E_1| - |\hat{E}| \end{cases} \tag{12}$$

Similarly, same equalities hold for G_2 and $\hat{G_2^j}$ for any $j \in \{1, \ldots, n_2\}$. Structural cost corresponding to edit path P is thus equal to:

$$\gamma_{struct}(P) = |V_1|c_{vd} + |V_2|c_{vi} + |E_1|c_{ed} + |E_2|c_{ei} - |\hat{V}|(c_{vd} + c_{vi}) - |\hat{E}|(c_{ed} + c_{ei}) \tag{13}$$

Computing substitution cost $\gamma_{label}(P)$ (equation 11) relies on computing the number of non identical node substitutions V_f and edge substitutions E_f transforming $\hat{G_1^i}$ into $\hat{G_2^j}$. Let $\Phi(\hat{G})$ denote the set of structural automorphisms of $\hat{G}$. Given both subgraphs $\hat{G_1^i}$ and $\hat{G_2^j}$, each automorphism $\phi \in \Phi(\hat{G})$ induces a mapping of $\hat{G_1^i}$ onto $\hat{G_2^j}$ and thus a substitution of the label of each vertex v (resp. edge e) of $\hat{G_1^i}$ onto the label of $\phi(v)$ (resp. $\phi(e)$) in $\hat{G_2^j}$. More precisely, let us denote by $P_{i,j,\phi}$ the edit path associated to the triplet $(\hat{G_1^i}, \hat{G_2^j}, \phi)$. the number of non identical substitutions V_f and E_f induced by $P_{i,j,\phi}$ is equal to:

$$\begin{aligned} V_f(P_{i,j,\phi}) &= |\{v \in \hat{V_1} \mid \hat{\mu}_1^i(v) \neq \hat{\mu}_2^j(\phi(v))\}| \\ E_f(P_{i,j,\phi}) &= |\{(v, v') \in \hat{E_1} \mid \hat{\nu}_1^i(v, v')) \neq \hat{\nu}_2^j(\phi(v), \phi(v'))\}| \end{aligned} \tag{14}$$

Substitution cost of edit path $P_{i,j,\phi}$ is thus equal to $\gamma_{label}(P_{i,j,\phi}) = V_f(P_{i,j,\phi})c_{ns} + E_f(P_{i,j,\phi})c_{es}$. Let us denote by P_{opt} the edit path minimizing the substitution cost:

$$P_{opt} = P_{i^\star, j^\star, \phi^\star} \text{ with } (i^\star, j^\star, \phi^\star) = \underset{(i,j,\phi) \in \{1,\ldots,n_1\} \times \{1,\ldots,n_2\} \times \Phi(\hat{G})}{\operatorname{argmin}} \gamma_{label}(P_{i,j,\phi}) \tag{15}$$

Since $\gamma_{struct}(P_{i,j,\phi})$ is the same for any $(i, j, \phi) \in \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \times \Phi(\hat{G})$ (equation 13), P_{opt} is an edit path having a minimal cost. Therefore, under our assumptions, the edit path associated to the edit distance is the one which passes through the pair of maximum common structural subgraphs and which minimizes the number of substitutions (equation 15).

This exact edit distance computation algorithm can be applied to treelets since the set of treelets is composed of 14 different structures. In addition, by restricting the set of edit paths to the ones which preserve the connectedness of intermediate graphs [26], we obtain a lower bound on the ratio between substitutions and insertion/deletion costs.

Proposition 3 *Considering edit paths preserving connectedness and given two trees T_1, $T_2 \in \mathcal{T}$, if $\frac{c_{vd}+c_{vi}}{c_{vs}} \geq \delta_v$ and $\frac{c_{ed}+c_{ei}}{c_{es}} \geq \delta_v - 1$, then $\hat{G_1}$ is a maximum common structural subtree of T_1 and T_2.*

Proof 3 *Can be found in [26].*

When computing tree edit distance on the set of treelets, δ_v is bounded by 6 and if we define costs as symmetric, i.e. $c_{vd} = c_{vi}$ and $c_{ed} = c_{ei}$, bounds on costs lead to: $c_{vd} > 3c_{vs}$ and $c_{ed} > 2.5c_{es}$. Since the set of treelets is defined as all trees having a size less than or equal to 6, the maximum common structural subtree of two treelets T_1 and T_2 is a treelet. The set of possible subgraphs and automorphisms for any pair of treelets can be easily pre computed since we have to consider only 14 structures. Therefore, computing the exact edit distance between two treelets consists in comparing at most $\max\limits_{(i,j)\in\{0,\dots,13\}^2}(n_i * n_j * |\Phi_{ij}|)$ label sequences where Φ_{ij} denotes the set of automorphisms of the maximum common structural subtree $\hat{T}$ of treelets T_i and T_j and n_i, n_j the numbers of subtrees of T_i and T_j isomorphic to $\hat{T}$. The value of this product on the set of treelets is bounded by 120, hence inducing a constant time complexity for the computation of the exact treelet edit distance. Note that, without our restriction to a set of specific tree structures, the complexity of the edit distance calculation between labeled unordered unrooted trees is NP-Complete [28].

Given a training set $D = \{G_0, \dots, G_N\}$ and a set of treelets $\{t_1, \dots, t_m\}$ extracted from graphs in D, we define a m by m similarity matrix corresponding to a gaussian RBF kernel applied on the exact graph edit distance:

$$K_{ted} = \left(e^{-\frac{d^2(t_i,t_j)}{\sigma_{inter}}}\right)_{(i,j)\in\{1,\dots,m\}^2} \tag{16}$$

Since graph edit distance does not define an euclidean matrix [7], matrix K_{ted} may not be semi definite positive and thus may not define a kernel on $\{t_1, \dots, t_m\}$. In order to define a valid kernel, we define a Gram matrix K on $\{t_1, \dots, t_m\}^2$ by regularizing K_{ted} :

$$K = K_{ted} - I\lambda_m \tag{17}$$

where λ_m corresponds to the lowest eigenvalue of K_{ted}. Note that this regularization has to be performed only once since this kernel only operates on treelets and not directly on graphs. In order to be applied on any test set, kernel k' between treelets is defined as:

$$k'(t,t') = \begin{cases} K_{(i,j)} \text{ if } t = t_i \in \{t_1, \dots, t_m\} \text{ and } t' = t_j \in \{t_1, \dots, t_m\}, \\ 1 \text{ if } t = t' \notin \{t_1, \dots, t_m\} \text{ and} \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

This kernel deduced from the 0-extension of K is trivially definite positive.

This kernel between treelets encodes a similarity measure between treelets based on an efficient computation of the graph edit distance. Using such a kernel allows us to include the contribution of non isomorphic but similar treelets. Considering the kernel k_{inter} defined in equation 8, k' associates isomorphic treelets to an high value since $d(t_i, t_j) = 0$ if $t_i \simeq t_j$. Conversely, highly dissimilar treelets are removed from kernel contribution since an high distance corresponds to a low value of gaussian kernel.

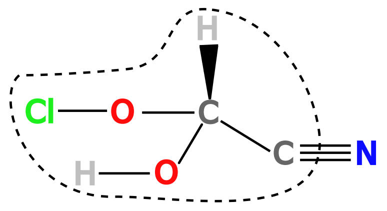
4 Including stereo information

As mentioned in section 1, stereoisomers correspond to molecules encoded by a same molecular graph which only differ by the relative positioning of their atoms. For example, considering figure 3(a) and looking at the central carbon (C) from the hydrogen atom (H), the remaining neighbors of C may be considered as lying on a plane and encountered clockwise. These remaining neighbors can thus be encoded by the sequence (C_N, O_H, O_{Cl}[1]) which identifies a particular stereoisomer whereas an opposite sequence of neighbors (e.g. (O_{Cl}, O_H, C_N)) corresponds to a distinct stereoisomer. This example is called an asymmetric carbon, a carbon with four different neighbours, and its presence is one of the most frequently encountered cause of stereoisomerism. Another cause of stereoisomerism is the presence of a double bond between carbons. According to chemical experts, 98% of the stereoisomers currently used in chemistry are stereoisomers because of the presence of one of this two configurations. Graph kernel proposed by [14] allows to encode these two forms of stereoisomerism by adapting the tree pattern kernel. However, in this approach, patterns which encode stereo information, and patterns which do not, are combined without any weighting in the final kernel value. Therefore, patterns which do not encode any stereo information may be assimilated to noise when predicting a property only related to stereoisomerism. In this section, we propose to adapt the treelet kernel to stereoisomers in order to obtain a kernel encoding only stereo information.

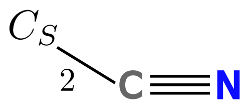
4.1 Ordered graphs and stereo vertices

Let us first consider asymmetric carbon. The spatial configuration of the stereoisomer depicted in figure 3(a) can be encoded by the sequence (H, C_N, O_H, O_{Cl}) where the atom fixing the point of view (H) and the first atom of the remaining neighbors cyclically ordered are arbitrary chosen. For each atom, the relative positioning of its neighbors may thus be encoded by an ordered list. An ordered graph $G = (V, E, \mu, \nu, ord)$ is defined as a molecular graph $G_m = (V, E, \mu, \nu)$ with a function $ord : V \to V^*$ which associates to each node an ordered list of its neighbors. Two ordered graphs G and G' are isomorphic ($G \underset{o}{\simeq} G'$) if it exists an isomorphism f between their respective molecular graphs G_m and G'_m such that $ord'(f(v)) = (f(v_1), \ldots, f(v_n))$ with $ord(v) = (v_1, \ldots, v_n)$.

[1] C_N denotes the atom C of the branch C−N, O_H the atom O of the branch O−H and O_{Cl} the atom O of the branch O−Cl.



(a) Molecule with one stereo vertex. Minimal stereo subtree is surrounded by a dashed line.



(b) Contracted graph obtained by contracting minimal stereo subtree into node C_S.

Figure 3: A stereoisomer with minimal stereo subtree and contracted graph representation.

The order encoding the neighborhood of a vertex being relative to the choice of two of its neighbors, two different ordering functions may encode a same configuration. Re-ordering functions associate to each vertex v of degree n a permutation on $\{1, \ldots, n\}$ which allows to re-order its neighborhood. The set of re-ordering functions encoding a same configuration is called a valid family Σ [29]. Two ordered graphs G and G' are considered as equivalent according to Σ (denoted $G \underset{\Sigma}{\simeq} G'$), if it exists a re-ordering function $\sigma \in \Sigma$ such that $\sigma(G) \underset{o}{\simeq} G'$. This relationship defines an equivalence relationship [29] and two different stereoisomers are encoded by non equivalent ordered graphs. This global graph's characterization may be translated into a local one in order to characterize vertex's stereo property:

Definition 1 (Stereo vertex) *Let $G = (V, E, \mu, \nu, ord)$ be an ordered graph. A vertex $v \in V$ of degree n is called a stereo vertex iff:*

$$\forall (i, j) \ \in \{1, \ldots, n\}^2, i \neq j, G \underset{\Sigma}{\not\simeq} \tau^v_{i,j}(G). \tag{19}$$

where $\tau^v_{i,j}(G)$ corresponds to an ordered graph deduced from G by permuting nodes of index i and j in $ord(v)$.

4.2 Minimal stereo subtree

Using definition 1, vertex's stereo property is characterized using the whole ordered graph G. Given a vertex v whose stereo property has to be checked by definition 1, one can observe that, on some configurations, the removal of some vertices far from v should not change its stereoisomerism. We should thus determine the minimal subgraph of G including v which fulfills equation 19, in order to obtain a local characterization of a stereo vertex. In the following, we restrict our attention to acyclic graphs. Considering such a restriction, the minimal subgraph characterizing the stereo property of a vertex v corresponds to the smallest ordered subtree rooted on v which allows to define v as a stereo vertex (definition 1). Such a subtree is called the minimal stereo subtree of v. Let us consider an ordered subtree T of G rooted on v and the set of subtrees $\{T_1, \dots, T_n\}$ of T rooted on each node adjacent to v, n being thus equals to the degree of v. Let us additionally consider the subtree T_i^j of T_i with the same root as T_i and whose depth is equal to j. We denote by $T(j_1, \dots, j_n)$ the subtree of T rooted on v and connected to the set of subtrees $\{T_1^{j_1}, \dots, T_n^{j_n}\}$. We then associate to each $i \in \{1, \dots, n\}$, the index $j_i^\star$ corresponding to the minimal value, and thus the minimal depth, such that $T_i^{j_i^\star}$ is not isomorphic to any T_k^j, $k \neq i, j \geq 1$. Any permutation of $T_i^{j_i^\star}$ and $T_k^{j_k^\star}$, $i \neq k$ in $T(j_1^\star, \dots, j_n^\star)$ leads thus to a non-isomorphic ordered tree and $T(j_1^\star, \dots, j_n^\star)$ is the minimal stereo subtree of v (equation 19). A molecule can thus be associated to a set of minimal stereo subtrees, each subtree being associated to a stereo vertex. Considering only acyclic graphs allows us to efficiently compare rooted ordered trees and thus minimal stereo subtrees. Indeed, an ordered tree is encoded by a string corresponding to the sequence of labels encountered during its depth first traversal. The canonical representation of an ordered tree, up to equivalence relationships is thus defined as the minimal string according to lexicographic order among all possible equivalent ordered trees (section 4.1). Ordered isomorphisms with equivalences between two trees can then be tested efficiently by comparing their unique string encodings [29].

4.3 Graph contraction and stereotreelet

We assume that the stereo properties of an acyclic molecule are determined both by its set of minimal stereo subtrees and by the relationships between these subtrees and the remaining parts of the molecule. These relationships may be encoded by a contracted graph (figure 3) where a minimal stereo subtree is contracted into a single node C_S. This contracted node is then connected to each leaf l of the corresponding minimal stereo subtree. Edges encoding these adjacency relationships are labeled by the sequence of child's indexes traversed to reach leaf l from the minimal stereo subtree's root.

Each stereo vertex v is associated to a contracted graph which encodes its minimal stereo subtree and the adjacency relationships of this tree with the other parts of the molecule. We then associate to v the set of treelets extracted from its contracted graph. This set of treelets is reduced to treelets containing the contracted node encoding the minimal stereo subtree. Each graph G can thus be associated to a set of stereotreelets $\mathcal{T}_S(G)$ defined as the union of stereotreelets associated to the stereo vertices of G. The stereotreelet kernel is then defined as a treelet kernel (equation 2) between the set of stereotreelets of both graphs to be compared:

$$k_{\mathcal{T}_S}(G, G') = \sum_{t \in \mathcal{T}_S(G) \cap \mathcal{T}_S(G')} k(f_t(G), f_t(G')) \qquad (20)$$

This extension defines a graph kernel which allows to distinguish two stereoisomers. Although it is restricted to acyclic molecules, this kernel allows to take into account local and non local chiral information. The extension of this framework to double bond between carbons is straightforward [29].

5 Including cyclic information

As mentioned in section 1, kernels based on linear or tree patterns do not encode any cyclic information while molecular cycles reduce the atom's degrees of freedom and have thus an high influence on molecular properties. Horváth et al. [13] have proposed the cyclic pattern kernel which compares the cycles of two graphs by computing their number of common simple cycles. However, the complexity required to enumerate the set of simple cycles is NP-Hard. Therefore, Horváth has proposed a new formulation [30] using relevant cycles [31] instead of simple cycles which reduces algorithm complexity. Nevertheless, this kernel deduces cyclic similarity from the number of common cycles but does not encode any adjacency relationships between cycles.

5.1 Relevant Cycles

Let us introduce relevant cycles, as defined by Vismara [31]. According to [31], a simple cycle is defined as a subgraph $C = (V', E', \mu, \nu)$ of a graph $G = (V, E, \mu, \nu)$ where each vertex $v \in V'$ has a degree equal to 2. Each cycle $C \trianglelefteq G$ can be represented as a vector $\vec{C} \in \{0,1\}^{|E|}$ where $\vec{C_i}$ is equal to 1 if i is an edge of C and to 0 otherwise. The set of vectors encoding cycles of G defines a vector space where the addition of two cycles C and C' corresponds to a XOR bitwise operation on their vectorial representations [31]. The set of relevant cycles of a graph, denoted $\mathcal{C}_{\mathcal{R}}(G)$, is defined as the union of all bases of the vector space of minimum length. The length of a base is defined as the sum of lengths of its cycles. Note that this set of cycles allows to encode all cycles of a molecular graph by combining them. Vismara has proposed an efficient algorithm which allows to compute the set of relevant cycles of a graph with a polynomial complexity according to the number of nodes of a graph [31].

5.2 Relevant Cycle Graph

Topological relationships between relevant cycles can be encoded by the relevant cycle graph. Given a graph $G = (V, E, \mu, \nu)$, its associated relevant cycle graph (figure 4) is defined as $G_{\mathcal{C}}(G) = (V_{\mathcal{C}}, E_{\mathcal{C}}, \mu_{\mathcal{C}}, \nu_{\mathcal{C}})$ where each vertex $v \in V_{\mathcal{C}}$ corresponds to a relevant cycle $c \in \mathcal{C}_{\mathcal{R}}(G)$. Each vertex $v \in V_{\mathcal{C}}$ is associated to the set $c_V(v) \subseteq V$ of nodes belonging to the cycle of G associated to v. Similarly, $c_E(v) \subseteq E$ encodes the set of edges composing the relevant cycle c associated to v. Two vertices $(u, v) \in V_{\mathcal{C}}^2$ are connected by an edge $(u, v) \in E_{\mathcal{C}}$ if their corresponding cycles share at least one node in G, i.e. if $c_V(u) \cap c_V(v) \neq \emptyset$. Labeling functions are defined as follows:

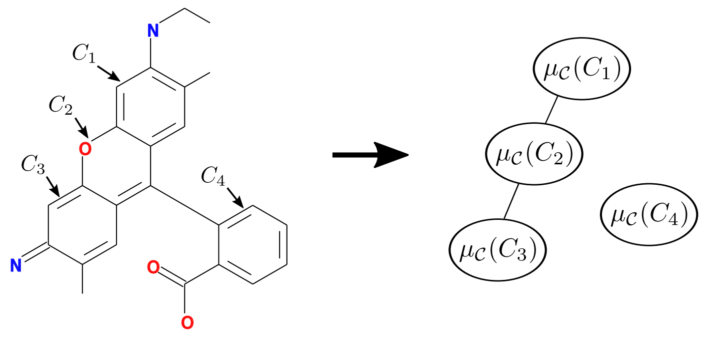


Figure 4: A molecular graph and its associated relevant cycle graph. Canonical key $\mu_{\mathcal{C}}(C_2)$ is equal to C1C1C1O1C1C2C.

- $\mu_{\mathcal{C}}(v)$: relevant cycle c encoded by v can be associated to a sequence of edge and vertex labels encountered during the traversal of c in G. In order to obtain a label sequence invariant to cyclic permutations, $\mu_{\mathcal{C}}(v)$ is defined as the sequence having the lowest lexicographic order;

- $\nu_{\mathcal{C}}(u,v)$: an edge $e = (u,v) \in E_{\mathcal{C}}$ is associated to each path shared by two cycles. Since this path can be traversed in two differents ways, we define $\nu_{\mathcal{C}}(e)$ as the minimal sequence of vertex and edge labels according to lexicographic order. Note that generally two relevant cycles share an unique path.

Cyclic information may be encoded into a graph kernel by applying treelet kernel on the relevant cycle graph instead of the original molecular graph. This kernel is thus defined by:

$$k_{\mathcal{C}}(G,G') = k_{\mathcal{T}}(G_{\mathcal{C}}(G), G_{\mathcal{C}}(G')) = \sum_{t_{\mathcal{C}} \in \mathcal{T}(G_{\mathcal{C}}(G)) \cap \mathcal{T}(G_{\mathcal{C}}(G'))} k(f_{t_{\mathcal{C}}}(G_{\mathcal{C}}(G)), f_{t_{\mathcal{C}}}(G_{\mathcal{C}}(G'))) \tag{21}$$

Where $k_{\mathcal{T}}$ refers to our treelet kernel (equation 2).

Using equation 21, extracted treelets no longer encode atom adjacency relationships but relevant cycle adjacency relationships. We can note that this kernel generalizes the cyclic pattern kernel. Indeed, the cyclic pattern kernel may be retrieved from the treelet kernel applied on the relevant cycle graph by restricting the set of treelets to the G_0 treelet structure (figure 1) and defining k as an intersection kernel. Considering others treelet structures allows us to encode adjacency relationships between relevant cycles.

However, we can note that kernel defined by equation 21 only includes cyclic information. Therefore, the cyclic pattern kernel, defined by equation 21, may be combined with a kernel encoding acyclic similarity such as treelet kernel applied on the original molecular graph:

$$k(G,G') = \lambda k_{\mathcal{T}}(G,G') + (1-\lambda) k_{\mathcal{C}}(G,G') \tag{22}$$

This kernel allows to encode both cyclic and acyclic information. Parameter λ allows to tune the trade-off between cyclic and acyclic information into kernel value computation. This parameter may be adapted according to a given property, by cross validation or multiple kernel learning (section 2.1.1). However, note that in this last case, sparsity constraint must be relaxed in order to insure that both subkernels are taken into account.

Kernel defined by equation 22 encodes a global similarity measure by a sum of two subkernels where each subkernel exclusively encodes cyclic or acyclic information. Although this approach allows to split acyclic and cyclic contributions, it does not allow to encode adjacency relationships between cyclic and acyclic parts of a molecule. For example, adjacency relationships between a cycle and its substituents (e.g. C_3 and N in figure 4) is not encoded by our kernel, neither by any graph kernel to the best of our knowledge.

5.3 Relevant Cycle Hypergraph

In order to encode adjacency relationships between cyclic and acyclic parts of a molecule, a first approach consists in adding vertices and edges corresponding to acyclic parts to our relevant cycle graph defined in section 5.2. Unfortunately, such an approach can not handle cases where an atom is connected to two distinct relevant cycles. As shown in figure 5(a), the atom labeled O is connected by an unique edge to two distinct cycles in the molecular graph representation. This adjacency relationship can not be encoded by a graph since an edge defines an adjacency relationship between two nodes. Therefore, in order to handle such relationships, we propose to define a new hypergraph representation encoding a molecular graph.

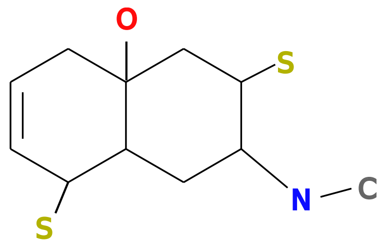
A directed hypergraph [32] $H = (V, E)$ can be defined as a set of vertices V and a set $E = E^e \cup E^h$ encoding the union of a set of edges $E^e \subset V \times V$ and a set of hyperedges $E^h \subset \mathcal{P}(V) \times \mathcal{P}(V)$ where $\mathcal{P}(V)$ encodes the set of all subsets of V. Note that unlike classical hypergraph's definitions, we explicitly distinguish the hyperedges from the classic edges. An ordered hyperedge $e = (s_u, s_v) \in E^h$ with $s_u = \{u_1, \dots, u_i\}$ and $s_v = \{v_1, \dots, v_j\}$ defines an adjacency relationship between node sets $\{u_1, \dots, u_i\}$ and $\{v_1, \dots, v_j\}$, as illustrated in figure 5(c). In the following, we assume that if there exists an $e \in E$ with $e = (s_1, s_2)$ then it exists $e' \in E$ with $e' = (s_2, s_1)$ and e and e' are considered as a same and unique hyperedge. Such a definition allows us to encode adjacency relationships between an acyclic atom and a set of cycles, each cycle being encoded as a vertex.

A molecular graph $G = (V, E, \mu, \nu)$ can be encoded as a relevant cycle hypergraph $H_{CH}(G) = (V_{CH}, E_{CH}, \mu_{CH}, \nu_{CH})$. Considering the relevant cycle graph $G_{\mathcal{C}}(G)$, the set of relevant cycles $C_{\mathcal{R}}(G)$ is associated to a set $V_{C_{\mathcal{R}}} \subseteq V$ corresponding to the union of all nodes included within a cycle, i.e. $V_{C_{\mathcal{R}}} = \{u \in c_V(v) \mid v \in V_{\mathcal{C}}\}$. Similarly, $E_{C_{\mathcal{R}}} = \{e \in c_E(v) \mid v \in V_{\mathcal{C}}\}$ encodes the set of edges included within a relevant cycle (section 5.2). The relevant cycle graph encodes all nodes and edges of a graph G which are included within a cycle. Therefore, molecular parts which are missing in the relevant cycle graph correspond to acyclic parts, i.e. nodes and edges which are not included within a cycle. These sets are respectively defined by the complement of $V_{C_{\mathcal{R}}}$, resp. $E_{C_{\mathcal{R}}}$, in V, resp. E. In order to encode both acyclic and cyclic parts into the relevant cycle hypergraph, V_{CH} is defined by the union of two subsets. A first subset $V_{\mathcal{C}}$ encoding the set of relevant cycles and a second subset $\{V \setminus V_{C_{\mathcal{R}}}\}$ corresponding to the set of atoms not included within a cycle.

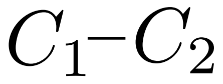
Considering the set of vertices V_{CH}, we can define a function $p : V \rightarrow \mathcal{P}(V_{CH})$ defined as $p(u) = \{u\}$ if $u \notin V_{C_{\mathcal{R}}}$ and $p(u) = \{v \in V_{\mathcal{C}} \mid u \in c_V(v)\}$ if not. The set $p(u)$ encodes either the cycles which include atom u or the atom itself. Similarly to vertices, the set of hyperedges E_{CH} is composed of two subsets:

1. A set of edges E^e_{CH} composed of:
   - edges between relevant cycle nodes which correspond to the set of edges $E_{\mathcal{C}}$ defined in the relevant cycle graph;
   - edges $e = (p(u), p(v))$ such that $(u, v) \in E \setminus E_{C_{\mathcal{R}}}$, $|p(u)| = 1$ and $|p(v)| = 1$. This set of edges corresponds to edges of the molecular graph G which encode three distinct adjacency relationships:
     - an edge connecting two acyclic atoms;
     - an edge between two single relevant cycles (for example C_2 and C_4 in figure 4);
     - or an edge connecting an acyclic atom of G to a single relevant cycle (C_3 and N in figure 4).
2. and a set of hyperedges $e = (p(u), p(v)) \in E^h_{CH}$ such that $(u, v) \in E \setminus E_{C_{\mathcal{R}}}$ and $|p(u)| > 1$ or $|p(v)| > 1$. This set of hyperedges encodes special cases depicted in figure 5 where an edge connects at least two distinct relevant cycles to another part of the molecule. This edge connects two sets of vertices $s_1 = p(u)$ and $s_2 = p(v)$ and is thus encoded by an hyperedge $e = (s_1, s_2) \in E^h_{CH}$.

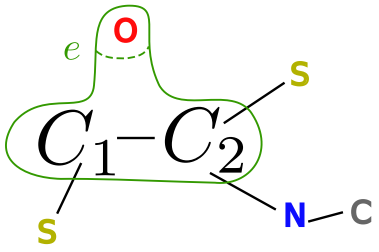
The labeling function μ_{CH} is equal to the vertex labeling function $\mu_{\mathcal{C}}$ of the relevant cycle graph (section 5.1) for each vertex $v \in V_{\mathcal{C}}$ corresponding to a relevant cycle, and to the vertex labeling function μ of the mocular graph otherwise. In the same way, the edge labeling function ν_{CH} is equal to the edge labeling function $\nu_{\mathcal{C}}$ of the relevant cycle graph for each edge encoding an adjacency relationship between two cycles and to the one of the molecular graph (ν) otherwise. Since hyperedges correspond to edges of the molecular graph connecting at least two distinct cycles to another part of the molecule, their labels are set as the ones of their associated edges in the molecular graph.



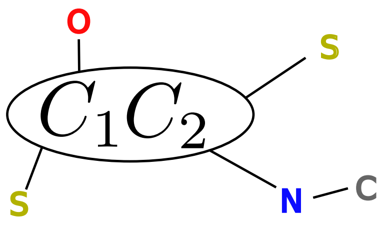
(a) Molecular graph G including cycles.



(b) Relevant cycle graph $G_{\mathcal{C}}(G)$.



(c) Relevant cycle hypergraph $H_{CH}(G)$. Hyperedge e encodes an adjacency relationship between atom O and C1 and C2.



(d) Reduced relevant cycle graph $G_{RC}(G)$.

Figure 5: Different encodings of a same molecule.

This molecular hypergraph representation (figure 5(c)) encodes all atoms $v \in V$ of the original molecular graph G either by a node encoding a cycle or by v itself if v is not included within any cycle. Similarly, each atomic bond encoded by an edge $e \in E$ in the molecular graph G is encoded in the molecular hypergraph representation. In addition, we note that the set of vertices incident to an hyperedge defines a clique:

Proposition 4 *Let be a graph $G = (V, E)$ and its associated relevant cycle hypergraph $H_{CH}(G) = (V_{CH}, E_{CH})$. If $\exists e = (s_1, s_2) \in E^h_{CH}$ and c_1, $c_2 \in V_{CH}$ such that $\{c_1, c_2\} \subseteq s_1$ or $\{c_1, c_2\} \subseteq s_2$, then $(c_1, c_2) \in E^e_{CH}$, i.e. c_1 is adjacent to c_2.*

Proof 4 *Without loss of generality, let us assume that $\{c_1, c_2\} \subseteq s_1$. Then by construction of E^h_{CH}, $\exists e = (u, v) \in E$ such that $\{c_1, c_2\} \subseteq p(u) = s_1$. By definition of function p and since $c_1, c_2 \in C_{\mathcal{R}}(G)$, it holds that $u \in c_V(c_1) \cap c_V(c_2)$. By definition of the relevant cycle graph, $(c_1, c_2) \in E_{\mathcal{C}} \subseteq E^e_{CH}$.*

5.4 Similarity between relevant cycle hypergraphs

Previous section defines a new molecular representation which allows to encode adjacency relationships between cyclic and acyclic parts of a molecule. In order to apply QSAR/QSPR methods using this molecular representation, we have to define a kernel between relevant cycle hypergraphs. An hypergraph encodes global relationships defined between sets of nodes. Conversely, treelet kernel is defined on graphs where relationships are defined locally between elementary nodes. Therefore, in order to apply treelet kernel to our hypergraph representation, we have to transform global relationships encoded in an hypergraph representation to local relationships between elementary nodes. This transformation may be performed by merging sets of vertices incident to an hyperedge. This merging operation relies to transform hyperedges to edges.

An equivalence relation $\sim$ between vertices $c \in V_{RH}$ is defined by the transitive closure of the relation R such that $c_1 \; R \; c_2$ if and only if it exists $e = (s_1, s_2) \in E_{CH}$ such that $\{c_1, c_2\} \subseteq s_1$ or $\{c_1, c_2\} \subseteq s_2$. Using equivalence relation $\sim$ previously defined, we can define the equivalence class $\bar{c} = \{c'; c \sim c'\}$ of a node $c \in V_{CH}$. Intuitively, two cycles sharing a common hyperedge belong to the same equivalence class. Then, by applying a contraction kernel on each class $\bar{c}$, we define a reduced relevant cycle graph (figure 5(d)) $G_{RC}(G) = (V_{RC}, E_{RC}, \mu_{RC}, \nu_{RC})$ associated to graph G with:

- $V_{RC} = \{\bar{c}, c \in V_{CH}\}$,
- $E_{RC} = \{e = (\bar{c_1}, \bar{c_2}), (c_1, c_2) \in E_{CH}, c_1 \nsim c_2\}$. Intuitively, the set of edges E_{RC} corresponds to the union of the usual edges E^{e}_{CH} of H_{CH} and the transformation of hyperedges E^{h}_{CH} into usual edges.

Labeling function $\mu_{RC}(\bar{c}), c \in V_{CH}$, is defined in a canonical way by the sequence of vertex and edge labels encountered during a depth first traversal of the spanning tree covering $\bar{c}$ and having the lowest lexicographic order. Such a spanning tree exists since any pair of vertices $\{c, c'\}$ sharing a same hyperedge is connected (Proposition 4).

Considering reduced relevant cycle graph, our new similarity measure based on treelet kernel is defined in two parts. A first step consists in extracting the set of treelets $\mathcal{T}_1 = \mathcal{T}(V_{CH}, E^{e}_{CH})$. The couple (V_{CH}, E^{e}_{CH}) corresponds to a subhypergraph of H_{CH} which does not include any hyperedge and which thus corresponds to a graph. The set of treelets $\mathcal{T}_1$ does not include special cases illustrated in figure 5. These special cases, corresponding to hyperedges $e \in E^{h}_{CH}$, are encoded by the set of treelets $\mathcal{T}_2$ which is defined as the set of treelets extracted from the reduced relevant cycle graph G_{RC}. In order to avoid redundancy, we reduce the set of treelets $\mathcal{T}_2$ to treelets containing at least one edge corresponding to an hyperedge $e_h \in E^{h}_{CH}$. Finally, we define the set of treelets $\mathcal{T}_{CR}(G)$ associated to a molecular graph G by $\mathcal{T}_1 \cup \mathcal{T}_2$. Our treelet kernel extension is then defined as a treelet kernel applied on the set of treelets $\mathcal{T}_{CR}$:

$$k_{RH}(G, G') = \sum_{t \in \mathcal{T}_{CR}(G) \cap \mathcal{T}_{CR}(G')} k(f_t(G), f_t(G')) \qquad (23)$$

where $k(.,.)$ is defined as a kernel between real numbers. Note that, multiple kernel learning methods defined in section 2.1 can be applied in order to extract relevant subtrees involving cycles and acyclic parts.

Table 1: Information encoded by each treelet's kernel extension.

Kernel	Acyclic	Non-isomorphic patterns	Stereo	Cyclic	Acyclic \Cyclic
(1) $k_{\mathcal{T}}$ (Equation 2)	✓				
(2) k_{inter} (Equation 8)	✓	✓			
(3) $k_{\mathcal{T}_{\mathcal{S}}}$ (Equation 20)	✓		✓		
(4) $k_{\mathcal{C}}$ (Equation 21)				✓	
(5) $k_{\mathcal{T}} + \lambda k_{\mathcal{C}}$ (Equation 22)	✓			✓	
(6) k_{RH} (Equation 23)	✓			✓	✓

The two new molecular representations defined in this section allows us to encode molecular cyclic information into our treelet kernel. Cyclic information can be taken into account separately from acyclic information by considering the relevant cycle graph or combined with acyclic information using the global molecular representation provided by the relevant cycle hypergraph. Prediction ability of each method depends on the chemoinformatics problem addressed by each dataset.

6 Experiments

We proposed several extensions of the treelet kernel in order to define a more accurate similarity measure for QSAR/QSPR problems. Table 1 summarizes the different informations took into account by each proposed extension. The first line corresponds to the treelet kernel $k_{\mathcal{T}}$ [11] which only takes into account acyclic information. Based on the hypothesis that similar sub structures should have a similar influence, the first extension (line 2) aims to include the comparison of non-isomorphic treelets into kernel computation. Similarity between treelets is deduced from graph edit distance. Second, the next extension (line 3) consists in encoding stereo information which is crucial for predicting certain molecular properties. Finally, the last three extensions (lines 4 to 6) consist in encoding cyclic information into graph kernels. First, we have proposed to apply the treelet kernel on the relevant cycle graphs (line 4) which allows us to encode the cyclic system of a molecule. However, considering such a kernel does not allow us to take into account acyclic parts. This drawback can be tackled by computing a kernel as a combination of a treelet kernel and a treelet kernel on relevant cycle graph (line 5). The parameter λ allows to weight the influence of the cyclic information and is choosen by cross-validation. However, this combination dissociates cyclic and acyclic parts and can't thus encode relationships between cyclic and acyclic parts of a molecule. Therefore, we proposed the relevant cycle hypergraph as a new molecular representation. This hypergraph representation allows us to encode the whole molecular graph and to explicitly encode cyclic information. We adapted the treelet kernel (line 6) in order to compare relevant cycle hypergraphs which defines thus a similarity measure

Table 2: Boiling point prediction of 185 acyclic molecules. Computation times are displayed in seconds.

Method	RMSE	learn.	pred.
(1) Path kernel	12.24 ± 1.45	7.83	0.18
(2) Random walks kernel	18.72 ± 1.74	19.10	0.57
(3) Tree pattern kernel	11.02 ± 1.06	4.98	0.03
(4) Treelet kernel	8.10 ± 0.88	**0.07**	**0.01**
(5) Inter treelet kernel with approx. edit distance	6.09 ± 0.6	12.43	0.07
(6) Inter treelet kernel with exact edit distance	5.89 ± 0.55	3.77	0.07
(7) Treelet kernel with MKL	**5.24** ± 0.53	70	**0.01**

encoding adjacency relationships between relevant cycles, acyclic parts, and between relevant cycles and acyclic parts. These contributions have been tested on three chemoinformatics datasets[2] in order to determine their relevancy from an experimental point of view. These experiments consists in two regression problems and one classification problem. Best parameters have been tuned by using a cross validation over the intervals $[10^{-2}, 10^4]$ for C (equation 6) and $[\sigma(y), \frac{\sigma(y)}{20}]$ for ϵ in SVM problems, where $\sigma(y)$ denotes the standard deviation of the predicted property. The values of σ_{inter} (equation 16) and σ range in $[0.1, 5]$ for RBF gaussian kernels and $\lambda \in [10^{-7}, 1]$ for kernel ridge regression. We also consider linear, gaussian, intersection and binary kernels as basic kernel between treelet's frequencies.

6.1 Boiling point prediction

The first prediction problem consists in predicting the boiling point of 185 acyclic molecules [2]. The results obtained using a kernel ridge regression with corresponding execution times in seconds are displayed in table 2. Column "learn." corresponds to the time required to compute the Gram matrix and column "pred." corresponds to the time required to predict the boiling point of a new molecule. We measure the accuracy of each method by computing the Root Mean Squared Error (column "RMSE") and confidence intervals at 95% (±). In this first experiment, we tested different graph kernels based on bags of patterns and two of our proposed extensions in order to show the gain obtained by the pattern weighting step and the comparison on non-isomorphic patterns. The two first lines of table 2 correspond to graph kernels based on linear patterns: the first line corresponds to a kernel based on paths extracted from the graphs [33] and the second line corresponds to the random walks kernel, as defined in [8]. Line 3 corresponds to the tree pattern kernel [9]. Line 4 corresponds to our treelet kernel (with $\lambda = 1$ and an intersection kernel) and lines 5 and 6 correspond to results obtained by including the comparison of

[2]All these datasets are available on the IAPR TC15 Web page: https://iapr-tc15.greyc.fr/links.html

similar patterns into treelet kernel computation using graph edit distance with $c_i = 3$ and $c_s = 1$ (section 3 and table 1, line 2). Best results have been obtained with an intersection kernel, $\lambda = 10^{-2}$ and $\sigma_{inter} = 0.5$ for line 5 and $\sigma_{inter} = 1$ for line 6. Finally, line 7 corresponds to the combination of the treelet kernel and MKL using SVM regression with $C = 10$, $\epsilon = 0.005$ and an intersection kernel (section 2.1.1).

The results show that the low expressiveness of linear patterns does not allow to predict accurately the boiling point of molecules conversely to kernels based on non linear patterns which obtain a more accurate prediction. In addition, we can note that the use of a limited set of structures allows to obtain low execution times. Then, we can note that the comparison of non-isomorphic patterns allows to obtain a better prediction accuracy than the treelet kernel, hence showing the relevance of including pairs of non-isomorphic treelets within kernel computation. In addition, we can note that the use of an exact edit distance provides a slightly better accuracy than using an approximate edit distance. Finally, best results are obtained thanks to the use of multiple kernel learning which allows to only consider relevant patterns at the cost of an higher computational time. However, note that prediction time is not altered by multiple kernel learning step since weighting is only computed during the learning step.

6.2 Optical rotation prediction

Table 3: Optical rotation angle prediction on acyclic chiral molecules.

Method	RMSE (°)	Time (s)
(1) Treelet kernel	26.24 ± 6.43	2
(2) Stereo tree-pattern kernel	24.16 ± 5.64	21
(3) Stereo treelet kernel	15.63 ± 3.54	0.4
(4) Treelet and stereo treelet kernel combination	**14.80** ± 3.30	2

The second regression experiment aims to show the relevancy of taking into account stereoisomerism. This experiment consists in predicting the optical rotation angle, which strongly depends on stereoisomerism properties, of 35 acyclic and chiral molecules by using a leave one out prediction and a SVM regression. Table 3 shows results obtained by the treelet kernel with $\epsilon = 20$, $C = 2$ and $\sigma = 0.5$ (line 1), tree pattern kernel adapted to stereoisomerism [14] (line 2), stereo treelet kernel (table 1, line 3) with $\epsilon = 10$, $C = 10^3$ and $\sigma = 4$ (line 3) and a combination of two kernels with $\epsilon = 5$, $C = 2$ and a binary kernel (line 4). This combination consists in predicting the sign of the optical rotation angle using the stereo treelet kernel and predicting its absolute value using the treelet kernel. On one hand, we can note that treelet kernel is not able to predict correctly this property since it can not distinguish two stereoisomers having nearly opposite optical rotation angles. On the other hand, stereo treelet kernel allows to distinguish stereoisomers and obtains a better prediction accuracy. The combination of the two kernels leads to the best results. Sign prediction step

Table 4: Classification accuracy on PTC dataset.

Method	# correct predictions				
	MM	FM	MR	FR	Time (s)
(1) Treelet kernel (TK)	208	205	<u>209</u>	212	27
(2) Cyclic pattern kernel	209	207	202	228	≤ 1
(3) TK on relevant cycle graph (TC)	211	210	203	232	≤ 1
(4) TK on relevant cycle hypergraph (TCH)	<u>217</u>	<u>224</u>	207	<u>233</u>	19
(5) TK + MKL	217	224	<u>223</u>	<u>250</u>	85
(6) TC + MKL	216	213	212	237	62
(7) TCH + MKL	**<u>225</u>**	<u>229</u>	215	239	117
(8) λ TK + (1 - λ) TCH	**225**	**230**	**224**	**252**	202

is performed without errors by stereo treelet kernel, hence showing the relevancy of this extension, and angle absolute value prediction performed by treelet kernel allows to slightly enhance the results obtained by the stereo treelet kernel alone. Therefore, patterns which include stereo information are useful to predict the sign of the optical rotation angle and patterns which do not are useful to predict its absolute value. As the tree pattern kernel adapted to stereoisomerism does not clearly identify the contribution of each kind of patterns, it can not obtain as good results as the stereo treelet kernel.

6.2.1 Predictive Toxicity Challenge

The third experiment aims to show the relevancy of explicitly including the cyclic information of molecules into kernel computation. The dataset used in this experiment is taken from the Predictive Toxicity Challenge [34] which aims to predict carcinogenicity of chemical compounds applied to four type of animals : female (F) and male (M) rats (R) and mice (M). This experiment consists in ten different datasets for each class of animal, each of them being composed of one training set and one test set. The amount of predicted molecules is equal to 336 for male mice, 349 for female mice, 344 for male rats and 351 for female rats. Table 4 shows the amount of correctly classified molecules over the ten test sets for each method and for each class of animal. In addition, the column time shows the learning time required by gram matrices computation and by the SVM learning step. First line of table 4 corresponds to original treelet kernel. The three next lines correspond to methods encoding three different levels of cyclic information. Line 2 corresponds to cyclic pattern kernel using relevant cycles [30]. Line 3 corresponds to treelet kernel applied on the relevant cycle graph (section 5.2 and table 1, line 4) which allows to encode relevant cycles relationships. Line 4 corresponds to the adaptation of treelet kernel to relevant cycle hypergraphs (section 5.3 and table 1, line 6) which allows to encode adjacency relationships between relevant cycles and acyclic parts. First, we can note that finer the cyclic information is encoded, better is the accuracy among the three methods explicitly encoding it (lines 2 to 4). These results validate our hypothesis on the importance of encoding adjacency relationships between relevant cycles in one hand and between relevant cycles and acyclic parts on the other hand. From a computational point of view, we can note that the low number of treelets extracted from the relevant cycle graph allows a fast learning step. In addition, the computational time required by the transformation of molecular graphs into relevant cycle graphs is balanced by the lower number of extracted treelets. The second part of table 4 (lines 5 to 8) shows results obtained by different kernels based on treelet kernel combined with a multiple kernel learning step as defined in section 2.1. Multiple kernel learning allows to significantly enhance the accuracy obtained by the different tested kernels (according to a Friedman test with a p-value equals to 0.01) at the price of an higher computational time. After this weighting step, treelet kernel adapted to relevant cycle hypergraph (line 7) obtains the best results on two datasets over four (datasets MM and FM) and kernel only based on acyclic patterns obtains the best results over the two others datasets (line 5, datasets MR and FR). This observation can be explained by the fact that some properties are influenced by cyclic information whereas some others depend more on acyclic information. Note that multiple kernel learning step selects about 25 relevant treelets among the 5700 extracted from relevant cycle hypergraphs. In comparison, treelet weighting step selects 150 treelets among 3500 when applied to molecular graph. The selected treelets consists of non linear treelets and treelets having 6 nodes which validates the use of non linear patterns composed of up to six nodes. Finally, treelet kernel on molecular graph has been combined with treelet kernel on relevant cycle hypergraph (table 4, line 8). This combination of two molecular representations obtains the best results on all datasets by taking the best combination of the two kernels for each dataset. As observed on the second part of results, best results are obtained for a λ equals to 0.9 for mice's datasets whereas best results are obtained for a λ equals to 0.1 for rats' datasets.

7 Conclusion

In this article, we have presented several extensions to treelet kernel which take into account chemical relevant characteristics such as pattern weighting, the contribution of non-isomorphic sub structures, stereoisomerism and cyclic information. As shown in experiments, these extensions allow to enhance prediction results obtained on several chemoinformatics datasets. In order to achieve a better prediction accuracy, we aim to encode the substituents' relative positioning around a cycle in order to encode a finer cyclic information. Another outlook consists in using optimal solvers such as MKL to define an inter treelet kernel based on a training set instead of using an *a priori* treelet edit distance. Finally, our next work regarding stereo information will focus on extending stereo treelet kernel to cyclic molecules in order to enlarge its application domain.

Acknowledgements

The authors would like to thank the computing center CRIHAN for providing the resources required by the computations of experiments.

References

[1] M. A. Johnson, G. M. Maggiora, Concepts and Applications of Molecular Similarity, Wiley, 1990.

[2] D. Cherqaoui, D. Villemin, A. Mesbah, J. M. Cense, V. Kvasnicka, Use of a neural network to determine the normal boiling points of acyclic ethers, peroxides, acetals and their sulfur analogues, Journal of Chemical Society Faraday Transactions 90 (1994) 2015–2019.

[3] T. Caelli, S. Kosinov, An eigenspace projection clustering method for inexact graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 515–519.

[4] P. Foggia, G. Percannella, M. Vento, Graph matching and learning in pattern recognition in the last 10 years, IJPRAI 28 (1).

[5] G. Poezevara, B. Cuissart, B. Crémilleux, Discovering emerging graph patterns from chemicals, in: ISMIS'2009, LNCS, Prague, 2009, pp. 45–55.

[6] K. Riesen, H. Bunke, Graph classification based on vector space embedding, IJPRAI 23 (6) (2009) 1053–1081.

[7] J. Dattorro, Convex optimization & Euclidean distance geometry, Meboo Publishing USA, 2005.

[8] H. Kashima, K. Tsuda, A. Inokuchi, Kernels for graphs, MIT Press, 2004, Ch. 7, pp. 155–170.

[9] P. Mahé, J.-P. Vert, Graph kernels based on tree patterns for molecules, Machine Learning 75 (1) (2008) 3–35.

[10] N. Shervashidze, K. M. Borgwardt, Fast subtree kernels on graphs, in: Advances in Neural Information Processing Systems, 2009, pp. 1660–1668.

[11] B. Gaüzère, L. Brun, D. Villemin, Two New Graphs Kernels in Chemoinformatics, Pattern Recognition Letters 33 (15) (2012) 2038–2047.

[12] N. Shervaszide, Scalable graph kernels, Ph.D. thesis, Universität Tübingen (2012).

[13] T. Horváth, T. Gärtner, S. Wrobel, Cyclic pattern kernels for predictive graph mining, in: KDD'2004, ACM Press, 2004, p. 158.

[14] J. Brown, T. Urata, T. Tamura, M. A. Arai, T. Kawabata, T. Akutsu, Compound analysis via graph kernels incorporating chirality, Journal of Bioinformatics and Computational Biology 8 (1) (2010) 63–81.

[15] B. Gaüzére, L. Brun, D. Villemin, Relevant cycle hypergraph representation for molecules, in: GBR'2013, Springer Berlin Heidelberg, 2013, pp. 111–120.

[16] P.-A. Grenier, L. Brun, D. Villemin, Treelet kernel incorporating chiral information, in: GBR'2013, Springer, 2013, pp. 132–141.

[17] M. Gönen, E. Alpaydın, Multiple kernel learning algorithms, Journal of Machine Learning Research 12 (2011) 2211–2268.

[18] D. Haussler, Convolution kernels on discrete structures, Tech. rep., Dept. of Computer Science, University of California at Santa Cruz (1999).

[19] N. Wale, I. Watson, G. Karypis, Comparison of descriptor spaces for chemical compound retrieval and classification, Knowledge and Information Systems 14 (3) (2008) 347–375.

[20] H. L. Morgan, The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service., Journal of Chemical Documentation 5 (2) (1965) 107–113.

[21] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet, SimpleMKL, Journal of Machine Learning Research 9 (2008) 2491–2521.

[22] M. Varma, D. Ray, Learning the discriminative power-invariance trade-off, in: ICCV'2007, IEEE, 2007, pp. 1–8.

[23] F. Yger, R. A., Wavelet kernel learning, Pattern Recognition 44 (10-11) (2011) 2614–2629.

[24] M. Neuhaus, H. Bunke, Bridging the Gap Between Graph Edit Distance and Kernel Machines, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.

[25] S. Fankhauser, K. Riesen, H. Bunke, Speeding up graph edit distance computation through fast bipartite matching, in: GBR'2011, Vol. 6658, 2011, pp. 102–111.

[26] L. Brun, B. Gaüzére, S. Fourey, Relationships between graph edit distance and maximal common unlabeled subgraph, Tech. rep., GREYC, http://hal.archives-ouvertes.fr/hal-00714879 (2012).

[27] H. Bunke, Error correcting graph matching: On the influence of the underlying cost function, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (9) (1999) 917–922.

[28] K. Zhang, R. Statman, D. Shasha, On the editing distance between unordered labeled trees, Information Processing Letters 42 (3) (1992) 133 – 139.

[29] P.-A. Grenier, L. Brun, D. Villemin, Incorporating chirality within the graph kernel framework, Tech. rep., CNRS UMR 6072 GREYC, http://hal.archives-ouvertes.fr/hal-00809066/ (2013).

[30] T. Horváth, Cyclic pattern kernels revisited, in: KDD'2005, Vol. 3518, 2005, pp. 791 – 801.

[31] P. Vismara, Union of all the minimum cycle bases of a graph, The Electronic Journal of Combinatorics 4 (1) (1997) 73–87.

[32] C. Berge, Graphs and hypergraphs, Vol. 6, Elsevier, 1976.

[33] F. Suard, A. Rakotomamonjy, A. Bensrhair, Kernel on bag of paths for measuring similarity of shapes, in: European Symposium on Artificial Neural Networks, 2002, pp. 355–360.

[34] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, C. Helma, Statistical evaluation of the predictive toxicology challenge 20002001, Bioinformatics 19 (10) (2003) 1183–1193.