A Polynomial-time Metric for Outerplanar Graphs

Leander Schietgat LEANDER.SCHIETGAT@CS.KULEUVEN.BE
Jan Ramon JAN.RAMON@CS.KULEUVEN.BE
Maurice Bruynooghe MAURICE.BRUYNOOGHE@CS.KULEUVEN.BE
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract

Graphs are mathematical structures that are capable of representing relational data. In the chemoinformatics context, they have become very popular for the representation of molecules. However, a lot of operations on graphs are NP-complete, so no efficient algorithms that can handle these structures exist. In this paper we focus on outerplanar graphs, a subclass within general graphs. Most molecular graphs are outerplanar. We define a metric on outerplanar graphs that is based on finding the maximal common subgraph and we present an algorithm that runs in polynomial time. Having an efficiently computable metric on molecules can improve the virtual screening of molecular databases significantly.

1. Introduction

Because of the huge costs of drug development, pharmaceutical companies have shown great interest in screening techniques, which automatically select from a database of molecules a set of candidates expressing a desired function. It is widely known that molecules with a similar structure tend to have the same function e.g., the binding capacity to some protein, so usually screening techniques will compare molecules by defining some kind of similarity measure between them. The task is then to search the database for molecules with a similar substructure as a given molecule.

During the last decades, molecular databases have grown enormously. This current evolution provides a great potential for the future of drug development, but there is a lack of techniques that are able to analyze this kind of databases efficiently.

Graphs[1] are mathematical structures that consist of a set of vertices and a set of edges, connecting the vertices. Graphs are excellent representations for binary relational data: a vertex can represent an entity, while an edge covers the relationship between two entities. In a graph describing a molecule for example, each atom is represented by a vertex and each atomic bond by an edge. Various variants exist to take into account labels of the vertices and the edges.

The representation of molecules by graphs has become very popular (Raymond & Willett, 2002). Unfortunately, searching for substructures in a graph is an instance of the subgraph isomorphism problem, which is NP-complete for general graphs. A typical approach to describe molecules is based on a chemical "fingerprint" (e.g., (Gillet et al., 1998)). In such cases, molecules are represented by a bit-string which lists the occurrence of a set of some predefined features, such as a ring structure or a functional group. This avoids the computational complexity of the subgraph isomorphism problem, but it ignores the underlying information about the molecule topology.

Previous work on graphs has shown that a lot of algorithms that are intractable for general graphs can be simplified when constraints are introduced on the structure of the graph. An example of a subclass of general graphs are outerplanar graphs. Outerplanar graphs are graphs which can be embedded in the plane in such a way that all of their vertices lie on the boundary of the outer face. (Horváth et al., 2006) have observed that 94.5% of the molecules in the NCI[2] database are outerplanar. This collection of datasets contains over 250000 chemical compounds which are used extensively for screening purposes. So having an efficient method to compare outerplanar graphs will result in significant speed-ups of molecule screening.

Proceedings of the 18th Benelearn
P. Adriaans, M. van Someren, S. Katrenko (eds.)
Copyright © 2007, The Author(s)

[1] For an overview on graph theory, see (Diestel, 2000).
[2] `http://cactus.nci.nih.gov/`.

In this paper we present a metric that computes a distance between outerplanar graphs in polynomial time. We could plug in this metric in a machine learning technique e.g., instance-based learning (Mitchell, 1997), which classifies objects on the basis of their nearest neighbors. Although our main application focuses on molecules, all of the domains where graphs are used to model the data (e.g., in image recognition and computer vision) would benefit from having an efficiently computable similarity measure between graphs.

The paper is organized as follows. In Section 2 we introduce the basic definitions. Section 3 gives a more formal description of the problem and shows an existing metric based on finding a maximal common subgraph. In Section 4 we discuss related work. Section 5 describes the representation of outerplanar graphs while Section 6 handles the algorithm for finding a maximal common subgraph of two outerplanar graphs. In Section 7 we draw conclusions and present future work.

2. Preliminaries

Definition 1 *A labeled graph is a quadruple $G(V, E, \Sigma, \lambda)$, with V a finite set of vertices and $E \subseteq \{e \subseteq V : |e| = 2\}$ a set of edges. Σ is a finite set of labels and $\lambda : V \cup E \rightarrow \Sigma$ is a function assigning a label to each element of $V \cup E$.*

Notation 1 *Let G be a graph. We denote with $V(G)$ the set of vertices of G, with $E(G)$ the set of edges of G and with λ_G the labeling function of G.*

Definition 2 *The open neighborhood of a vertex v in a graph G, denoted $N_G^0(v)$, is the set of all the vertices adjacent to v, i.e. $N_G^0(v) = \{x \mid \{v, x\} \in E(G)\}$. The closed neighborhood of v, denoted $N_G(v)$, is the set of v and all its neighbors, i.e. $N_G(v) = N_G^0(v) \cup \{v\}$.*

Definition 3 *In a graph G, a sequence $x_0, x_1, \ldots, x_n$ of vertices is a path from x_0 to x_n iff $\{x_i, x_{i+1}\} \in E(G)$, for all $i \in [0, n-1]$.*

Definition 4 *A cycle $x_0, \ldots, x_n$ is a path such that $x_0 = x_n$. A graph is acyclic if the graph contains no cycles.*

Definition 5 *A path without repeated vertices is a simple path; a cycle without repeated vertices apart from the start and end vertex is a simple cycle.*

Definition 6 *A graph G is connected if there is a path between any pair of its vertices; it is biconnected if for any two vertices u and v of G, there is a simple cycle containing u and v.*

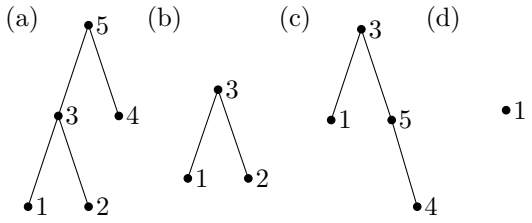


Figure 1. (a) T^5 (b) T_5^3 (c) T_2^3 (d) T_5^1

Definition 7 *A tree T is a graph for which there is a unique path between every pair of vertices[3] $u, v \in T$.*

Notation 2 *As edges are undirected, any vertex can be the root of a tree. When we choose to use vertex r as the root of a tree T we denote it as T^r. $Ch_{T^r}(v)$ denotes the set of children of vertex v in T^r. By T_i^r we denote the maximal subtree of T^r for which the two following properties hold: (i) $r \in T_i^r$ (the root) and (ii) if e is an edge of the path between r and i then $e \notin E(T_i^r)$.*

Example 1 *Figure 1 (a) shows a tree T where node 5 is selected as root, i.e., T^5. Figure 1 (b) shows T_5^3, the same tree, but now rooted at 3 and with edge $\{3, 5\}$ removed, which also results in the removal of $\{5, 4\}$. Figure 1 (c) shows T_2^3 and Figure 1 (d) shows T_5^1. Notice that $T^5 = T_5^5$.*

Definition 8 *Let G and H be graphs. G is a subgraph of H, if (i) $V(G) \subseteq V(H)$, (ii) $E(G) \subseteq E(H)$, (iii) G is connected, and (iv) $\lambda_G(x) = \lambda_H(x)$ holds for every $x \in V(G) \cup E(G)$.*

Definition 9 *The graph G of a graph H induced by a set of vertices $V \subseteq V(H)$ is defined as (i) $V(G) = V$, (ii) $E(G) = \{\{u, v\} \mid \{u, v\} \in H \wedge u, v \in V\}$, and (iii) $\lambda_G(x) = \lambda_H(x)$.*

Definition 10 *Two graphs G and H are isomorphic if there exists a bijection $\varphi : V(G) \rightarrow V(H)$ such that (i) $\{u, v\} \in E(G)$ iff $\{\varphi(u), \varphi(v)\} \in E(H)$, (ii) $\lambda_G(u) = \lambda_H(\varphi(u))$, and (iii) if $\{u, v\} \in E(G)$ then $\lambda_G(\{u, v\}) = \lambda_H(\{\varphi(u), \varphi(v)\})$ hold for every $u, v \in V(G)$.*

Notation 3 *We denote the set of all graphs with $\mathcal{G}$. Isomorphism is an equivalence relation on $\mathcal{G}$. We will*

[3] In the context of trees, vertices are also called nodes.

denote the set of all equivalence classes under isomorphism with $\mathcal{G}^{\equiv}$.

Definition 11 *A graph G is subgraph isomorphic to H, denoted $G \preceq H$, iff G is isomorphic to a subgraph of H. We write $G \preceq_{\varphi} H$ when we want to make the bijection φ explicit.*

Definition 12 *A size $|\cdot| : \mathcal{G} \to \mathbb{R}$ is a function mapping a graph to a real number of the form $|G| = \sum_{x \in V(G) \cup E(G)} w_{\lambda_G(x)}$, where each possible label of $l \in \Sigma$ has been assigned a weight w_l.*

Example 2 *Assume that all vertices of $v \in V(G)$ have the same label $\lambda_G(v) = l_1$ and all the edges $e \in E(G)$ have the same label $\lambda_G(e) = l_2$. Let $w_{l_1} = 1$ and $w_{l_2} = 0$. Then, the corresponding size function maps every graph G on the number of vertices of G.*

Definition 13 *A simple curve is the image of an injective continuous function γ: $[0,1] \to \mathbb{R}^2$; its endpoints are $\gamma(0)$ and $\gamma(1)$. Notice that by definition, simple curves are non self-intersecting. Let G be a graph. An embedding of G in the plane is a function s mapping each vertex of G to a distinct point of the plane and each edge $\{u, v\}$ of G to a simple curve of the plane connecting $s(u)$ and $s(v)$. If, in addition, it holds that any two distinct curves representing edges do not intersect except possibly at their endpoints then s is a planar embedding of G. A graph is planar if it has a planar embedding.*

Definition 14 *Let G be a planar graph and s be some planar embedding of G. Removing from the plane all points and curves corresponding to the vertices and edges of G, respectively, we obtain a set of connected pieces of the plane, called faces. Since the number of vertices is finite, exactly one of the faces, called the outer face, is unbounded.*

Definition 15 *An outerplanar graph is a planar graph which can be embedded in the plane in such a way that all of its vertices lie on the boundary of the outer face. We will call such an embedding an outerplanar embedding. We will denote the set of all outerplanar graphs with $\mathcal{G}_{op}$ and the set of its isomorphism classes with $\mathcal{G}_{op}^{\equiv}$.*

Definition 16 *A connected component of a graph G is a maximal subgraph of G that is connected. A biconnected component or block of a graph G is a maximal subgraph of G that is biconnected.*

Definition 17 *A bridge is an edge that does not belong to a block.*

Definition 18 *A block and bridge preserving (BBP) subgraph isomorphism from G to H is a subgraph isomorphism from G to H mapping (i) the set of bridges of G to the set of bridges of H and (ii) different blocks of G to different blocks of H.*

Notation 4 *We denote that a graph G is BBP subgraph isomorphic to H by $G \sqsubseteq H$.*

The subgraph isomorphism problem, which tries to find a subgraph isomorphism between two graphs is NP-hard (Garey & Johnson, 1979), even if restricted to the class of outerplanar graphs (Syslo, 1982). On the contrary, the BBP subgraph isomorphism problem is computable in polynomial time for outerplanar graphs (Horváth et al., 2006). For trees, which are special outerplanar graphs (i.e., block-free), the BBP subgraph isomorphism is equivalent to the subtree isomorphism. From an application point of view, the BBP subgraph isomorphism can be motivated from the fact that in molecules cyclic structures and linear fragments usually behave differently, and hence treating them separately is not necessarily a bad thing.

Definition 19 *A maximal common subgraph (MCS) I of two graphs G and H is a graph for which $I \preceq G$, $I \preceq H$ and there exists no other subgraph J for which $I \preceq J$.*

The time complexity of the MCS problem, which finds an MCS between two arbitrary graphs, is NP-hard as well (Garey & Johnson, 1979). The MCS problem for two trees can be solved in polynomial time (Garey & Johnson, 1979).

Definition 20 *A matching f between sets A and B is a relation such that for all $(a_1, b_1), (a_2, b_2) \in f$: $a_1 = a_2$ iff $b_1 = b_2$, so each element of A is associated with at most one element of B and vice versa.*

Definition 21 *A weighted maximal matching problem, also known as the assignment problem, is an optimisation problem where two sets A and B are given together with a weight function $w : A \times B \to \mathbb{R}$ and the task is to find a matching m between A and B such that $\sum_{(a,b) \in m} w(a, b)$ is maximal.*

Definition 22 *A pseudo-metric on a set Ω is a function $d : \Omega \times \Omega \to \mathbb{R}$ that fulfills three requirements:*

- *reflexivity:* $\forall x \in \Omega : d(x, x) = 0$,
- *symmetry:* $\forall x, y \in \Omega : d(x, y) = d(y, x)$,
- *triangle inequality:* $\forall x, y, z \in \Omega : d(x, z) \leq d(x, y) + d(y, z)$.

If it also holds that $\forall x, y \in \Omega : d(x, y) = 0 \Rightarrow x = y$, *then* d *is called a metric.*

3. Problem description

The goal of this paper is to develop a metric on $\mathcal{G}_{op}^{\equiv}$ which induces an efficiently computable pseudo-metric on $\mathcal{G}_{op}$.

(Bunke & Shearer, 1998) proposed a distance function on graphs based on the maximal common subgraph:

$$d_{bs}(G, H) = 1 - \frac{|MCS(G, H)|}{\max(|G|, |H|)},$$

with $|G|$ equal to the number of vertices in G. They proved that d_{bs} is a metric. We can easily extend this proof to the more general case where $|G|$ is determined by the function $size$.

However, this metric is not practical since computing the MCS of two general graphs is not possible in polynomial time in the size of the graphs (unless $P = NP$). To have a practical metric, we need a polynomial algorithm to find the size of the MCS of two outerplanar graphs. Therefore it will be necessary to redefine the MCS in function of the BBP subgraph isomorphism: we say that a maximal common connected subgraph under BBP subgraph isomorphism I of two outerplanar graphs G and H is a maximal connected graph for which $I \sqsubseteq G$ and $I \sqsubseteq H$. Throughout this paper, we will just refer to it as the MCS, unless stated otherwise.

4. Related work

Previous research on similarity measures for molecules has resulted in a large number of different approaches. This section briefly discusses the most important ones.

Feature-based distances use a vector-representation to reflect a set of predefined features a molecule has, i.e. 1 if the feature occurs in the molecule, 0 otherwise. The distance between two molecules is then reduced to the distance between the two corresponding bit-strings, which is efficiently computable. This solves the computational complexity, but it is difficult to decide which features to use.

Since feature-based distances do not take into account the actual topological structure, a new approach, the so called *cost-based distances*, has emerged. Here, the similarity between two molecule graphs is based on graph edit distances[4]. (Bunke, 1997) proves that techniques which compute the MCS of two graphs are strongly related to graph edit distances.

(Raymond & Willett, 2002) give an elaborate overview of existing similarity measures that are specifically graph-based. They also try to construct a methodology which assigns the algorithms into different categories. First, they make a distinction between approximation algorithms and exact algorithms. Approximation algorithms try to avoid the computational complexity by using several heuristics. As we are only interested in having an exact size of the MCS, we do not elaborate on these techniques. Second, some algorithms find the connected MCS, while others can also give an unconnected MCS as solution. Our approach only considers connected subgraphs, but it could be adapted to the unconnected case.

It turns out that in the existing literature, there are many definitions for the MCS. Another distinction has to be made between the Maximal Common Induced Subgraph (MCIS) and the Maximum Common Edge Subgraph (MCES). The MCIS requires the common subgraph to be induced. The MCES, which does not have this requirement, tries to maximize the number of edges in the common subgraph. Chemists have argued that the MCES more adequately expresses the notion of chemical similarity than does the MCIS since the bonds between atoms in a molecule are the most responsible for its activity. Since our definition of the MCS is defined in function of the BBP subgraph isomorphism, it is different from both the MCIS and the MCES. However, our approach mostly resembles the latter, since our definition of a subgraph is not the induced version.

In the chemoinformatics context, the current most important algorithms which solve the MCS problem are based on finding a maximum clique in the corresponding modular product graph of the two input graphs. (Raymond et al., 2002) propose a multi-step algorithm which theoretically still belongs to the class NP, but it makes use of some advanced heuristics to speed up the search. A somewhat older approach is based on a backtracking algorithm (e.g., (McGregor, 1982)) that uses various heuristics in order to reduce the number of backtrackings.

(Akutsu, 1993) proposed a polynomial algorithm for computing the MCS of "almost trees of bounded degree" under the general subgraph isomorphism. This is another class of graphs which can also be used for the representation of molecules. It would be interesting to have a comparison between this approach and

[4]The edit distance of two graphs G and H is the shortest (or least cost) sequence of edit operations that transform G into H. Examples of such operations are inserting, deleting and substituting a vertex or an edge.

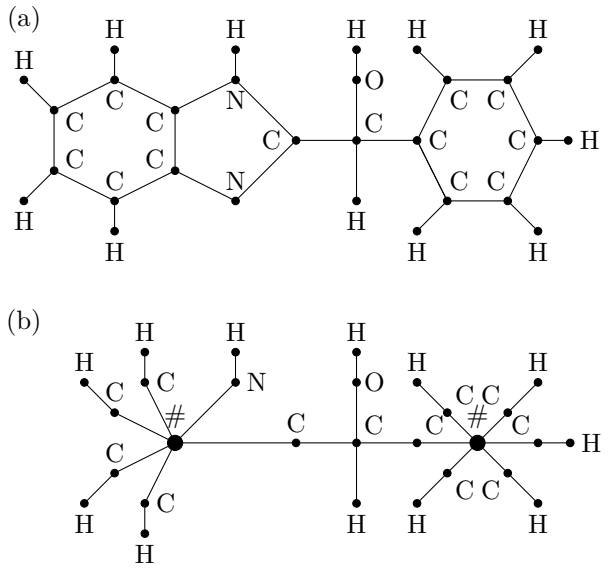


Figure 2. (a) An outerplanar graph. (b) Its corresponding block-bridge tree. Note that 3 vertices, internal to the left block, do not occur in this representation.

ours.

5. Families of partial graph structures

In this section, we define two kinds of partial graph structures: the non-block-splitting subgraphs and the half-graphs. These subgraphs will be used as auxiliary structures for the algorithm.

Recently, (Horváth et al., 2006) presented a tree representation for outerplanar graphs, called "block-bridge" trees. To transform an outerplanar graph G into its corresponding block-bridge tree, do the following for each block B in G: (i) add a new vertex labeled by #, (ii) remove each edge belonging to B and (iii) for every vertex v of B, if v is adjacent to a bridge or to another block of G, then connect v with v_B by an edge labeled by #; otherwise remove v. As an example, Figure 2 (a) shows an outerplanar graph and (b) shows its block-bridge tree representation. Edge labels are not shown.

Let T be the block-bridge tree corresponding to a graph G. We will denote with $\mathcal{T}(G)$ the set of all trees T_i^r where i and r is a vertex of T and r is not a vertex of T labeled by # (corresponding to a block). Then we define $G|T_i^r$ to be the subgraph of G induced by all vertices in T_i^r, including the ones that represent a block. In this way we can recover the vertices from the original graph which were lost in the tree representation. We will call $G|T_i^r$ a non-block-splitting subgraph of G, rooted at r. We also define $\mathcal{T}^*(G) = \{G|T_i^r \mid T_i^r \in \mathcal{T}(G)\}$.

We will now introduce the family of half-graphs of a graph G. An embedding of a block of an outerplanar graph has a unique simple (or Hamiltonian) cycle (Mitchell, 1979). We can either number the block's vertices clockwise ($\curvearrowright$) or counterclockwise ($\curvearrowleft$). An orientation is an element from the set $\{\curvearrowleft, \curvearrowright\}$. In the context of a block B, if $o \in \{\curvearrowleft, \curvearrowright\}$ is an orientation and $u \in V(B)$ is a vertex, we denote with $u + o(i)$ the vertex going i steps in the orientation o (either clockwise or counterclockwise). If $u, v \in V(B)$ are distinct vertices, we will denote with $o[u, v]$ the sequence of vertices along the Hamiltonian path of B between (and including) u and v. Notice that $o[u, v]$ is uniquely defined even without mentioning the block as a pair of vertices can only have at most one common block. We call such a sequence $o[u, v]$ a block interval. We denote the set of all block intervals of G with $\mathcal{I}nt(G)$. We say that $o[x, y] \subseteq o[u, v]$ iff $x, y \in o[u, v]$ and x comes before y in $o[u, v]$, according to the orientation o.

Let G be a graph and let $u, v \in V(B)$ be distinct vertices of a common block. We define the half-graph $G|_{o[u,v]}$ to be the maximal connected subgraph of G containing all vertices of $o[u, v]$ but none of the vertices $V(B) \setminus o[u, v]$ and none of the edges adjacent to v, which do not belong to the block B. We only consider half-graphs of at least 3 nodes. Using this notation, one can see that $G|_{o[u,v]} \cup G|_{o[v,u]} = G$. Moreover, if $\mathcal{I}_1, \mathcal{I}_2 \in \mathcal{I}nt(G)$ and $\mathcal{I}_1 \subseteq \mathcal{I}_2$, we have $G|_{\mathcal{I}_1} \preceq G|_{\mathcal{I}_2}$.

Example 3 *Figure 3 shows an outerplanar graph G, in which the half-graph $G|_{\curvearrowright[u,v]}$ is highlighted in bold. Notice that its complement is equal to $G|_{\curvearrowright[v,u]}$.*

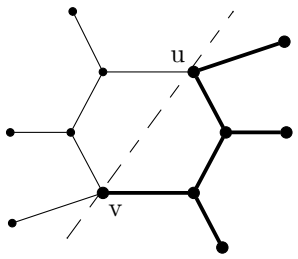


Figure 3. A half-graph $G\,|_{\curvearrowright[u,v]}$.

As we are using the BBP subgraph isomorphism, we introduce an extra edge to avoid the edges of blocks becoming bridges when constructing $G|_{o[u,v]}$. We define $G|^*_{o[u,v]}$ to be the graph $G|_{o[u,v]}$ where an edge $\{u, v\}$ with a special label \$ has been added if $G|_{o[u,v]}$ did not contain an edge between u and v itself (with $w_\$ = 0$).

We also define $\mathcal{I}nt^*(G) = \{G|^*_{\mathcal{I}} \mid \mathcal{I} \in \mathcal{I}nt(G)\}$.

Let G and H be two graphs. We say that $H \sqsubseteq^* G$ iff either $H \sqsubseteq G$ or there is an outerplanar graph G' which can be obtained from G by adding one edge with label \$ and $H \sqsubseteq G'$. One can show that if $o[x,y] \subseteq o[u,v]$, then $G|^*_{o[x,y]} \sqsubseteq^* G|^*_{o[u,v]}$.

Finally, we define for two outerplanar graphs G and H the set $\mathcal{P}(G,H)$ containing all possible couples of non-block-splitting subgraphs and half-graphs:

$$\mathcal{P}(G,H) = (\mathcal{T}^*(G) \times \mathcal{T}^*(H)) \cup (\mathcal{I}nt^*(G) \times \mathcal{I}nt^*(H)).$$

6. A dynamic programming algorithm

Our algorithm to compute the size of the maximal common connected subgraph under BBP subgraph isomorphism of two given outerplanar graphs G and H is based on a dynamic programming strategy. First, partial solutions are computed for pairs of smaller structures, and then building on these, the size of the maximal common subgraphs is computed for structures of increasing size, until the size of the maximal common subgraph of G and H themselves is computed.

Let G and H be two outerplanar graphs embedded in the plane, let $G|T^r_i \in \mathcal{T}^*(G)$ and let $H|U^s_j \in \mathcal{T}^*(H)$ be two non-block-splitting subgraphs. Then, we define

$$\sigma(G|T^r_i, H|U^s_j) = \max\{|S| \mid \exists \varphi_G, \varphi_H : S \sqsubseteq_{\varphi_G} G|T^r_i \wedge S \sqsubseteq_{\varphi_H} H|U^s_j \wedge \varphi_G(\varphi_H^{-1}(s)) = r\}.$$

In words, σ contains the size of the MCS of the two graphs $G|T^r_i$ and $H|U^k_j$ given that their root vertices (which cannot belong to blocks) are mapped onto each other.

Let $o^G[v^G_1, v^G_2] \in \mathcal{I}nt(G)$ and $o^H[v^H_1, v^H_2] \in \mathcal{I}nt(H)$. We now define $\rho(o^G[v^G_1, v^G_2], o^H[v^H_1, v^H_2])$ to be the set of all graphs S for which $S \sqsubseteq^*_{\varphi_G} G|^*_{o^G[v^G_1, v^G_2]}$ and $S \sqsubseteq^*_{\varphi_H} G|^*_{o^H[v^H_1, v^H_2]}$ for some φ_G and φ_H such that $\varphi_G(\varphi_H^{-1}(v^H_1)) = v^G_1$ and $\varphi_G(\varphi_H^{-1}(v^H_2)) = v^G_2$. In words, $\rho(\mathcal{I}_G, \mathcal{I}_H)$ represents the set of common subgraphs under the BBP subgraph isomorphism of $G|^*_{\mathcal{I}_G}$ and $H|^*_{\mathcal{I}_H}$ such that the endpoints of the block intervals $\mathcal{I}_G$ and $\mathcal{I}_H$ correspond to each other. We also define the function:

$$\sigma(G|^*_{\mathcal{I}_G}, H|^*_{\mathcal{I}_H}) = \max_{S \in \rho(\mathcal{I}_G, \mathcal{I}_H)} |S|$$

for every $\mathcal{I}_G \in \mathcal{I}nt(G)$ and $\mathcal{I}_H \in \mathcal{I}nt(H)$. σ contains again the size of the MCS of $G|^*_{\mathcal{I}_G}$ and $H|^*_{\mathcal{I}_H}$.

We have defined a function σ from $\mathcal{P}(G,H)$ to the set of real numbers. We will now follow a dynamic programming approach to compute all numbers $\sigma(p)$ for every pair p of $\mathcal{P}(G,H)$. As we want to make sure we process all $p \in \mathcal{P}(G,H)$ in "increasing size", we will first define an ordering relation on $\mathcal{P}(G,H)$.

Let $(p_G, p_H), (p'_G, p'_H) \in \mathcal{P}(G,H)$. We define $(p_G, p_H) \leq (p'_G, p'_H)$ iff $p_G \sqsubseteq^* p'_G$ and $p_H \sqsubseteq^* p'_H$. It is easy to order $\mathcal{P}(G,H)$ according to this order, as it is sufficient to order the pairs $(p_G, p_H) \in \mathcal{P}(G,H)$ lexicographically according to $(|p_G|, |p_H|)$.

Now we are ready to consider Algorithm 1. This main loop makes further calls to the specific matching functions depending on the nature of what should be matched in every step.

Algorithm 1 FIND_MCS(G,H): FINDING THE MCS OF TWO OUTERPLANAR GRAPHS

Require: G and H are outerplanar graphs.
Ensure: |MCS| is the size of the maximal common connected subgraph under BBP subgraph isomorphism of G and H.

```
1: for all (p_G, p_H) ∈ P(G,H) in increasing order do
2:     if p_G ∈ T*(G) ∧ p_H ∈ T*(H) then
3:         MatchVertex(p_G, p_H)
4:     else
5:         MatchBlock(p_G, p_H)
6:     end if
7: end for
8: return max_{(p_G,p_H)∈P(G,H)} σ(p_G, p_H)
```

One can show that this algorithm works correctly by verifying for every assignment of a value to $\sigma(p_G, p_H)$ that this value is a correct increment of the values of $\sigma(p'_G, p'_H)$ used (where $(p'_G, p'_H) \leq (p_G, p_H)$). Due to space restrictions, we omit a full proof. Instead, we will give a high-level explanation.

For the matching of two non-block-splitting subgraphs, the procedure MATCHVERTEX compares the labels of their root nodes (which will never belong to a block). If one of these subgraphs consists of a single vertex, the labels of both roots are compared[5]. If both roots have children in the block-bridge tree (they consist of multiple vertices), then a weighted maximal matching is computed between these children. The weights are calculated by the function MATCHEDGE, which considers two cases. Firstly, if both children refer to "regular" vertices in the block-bridge tree, then we can make use of the earlier computed solution for the non-block-splitting subgraphs which have the children as root. Secondly, if both children refer to blocks in the

[5]Recall from Definition 12 that every label $\lambda(x)$ has been assigned a weight $w_{\lambda(x)}$. $\delta_{\lambda(x),\lambda(y)}$ is $w_{\lambda(x)}$ if $\lambda(x) = \lambda(y)$, 0 otherwise.

Algorithm 2 Computing $\sigma(G|T_i^r, H|U_j^s)$

1: **procedure** MATCHVERTEX($(G|T_i^r, H|U_j^s)$)
2: **if** ($|V(G|T_i^r)| = 1$) or ($|V(H|U_j^s)| = 1$) **then**
3: $\sigma(G|T_i^r, H|U_j^s) \leftarrow \delta_{\lambda(r),\lambda(s)}$
4: **else if** $\lambda(r) = \lambda(s)$ **then**
5: $M \leftarrow Matching(Ch_{T_i^r}(r), Ch_{U_j^s}(s))$
6: $\sigma(G|T_i^r, H|U_j^s) \leftarrow w_{\lambda(r)} +$ $\max_{m \in M} \sum_{(x,y)\in m}$ MATCHEDGE$(G|T_i^r, x, H|U_j^s, y)$
7: **else**
8: $\sigma(G|T_i^r, H|U_j^s) \leftarrow 0$
9: **end if**
10: **end procedure**
11: **function** MATCHEDGE($G|T_i^r$, x, $H|U_j^s$, y)
12: **if** x refers to a vertex of G ($\lambda_{T_i^r}(x) = \#$) and y refers to a vertex of H ($\lambda_{U_j^s}(y) = \#$) **then**
13: **return** $\sigma(G|T_i^x, H|U_j^y).\delta_{\lambda(\{r,x\}),\lambda(\{s,y\})}$
14: **else if** x refers to a block B_G of G and y refers to a block B_H of H **then**
15: $N_r \leftarrow N_G^0(r) \cap V(B_G)$
16: $N_s \leftarrow N_H^0(s) \cap V(B_H)$
17: $maxval \leftarrow 0$
18: **for all** $r' \in N_r$, $s' \in N_s$ and $o^G, o^H \in \{\curvearrowleft, \curvearrowright\}$ **do**
19: **if** $\{r, r'\} \in E(G)$ and $\{s, s'\} \in E(H)$ and $\lambda_G(\{r, r'\}) = \lambda_H(\{s, s'\})$ **then**
20: $val \leftarrow \sigma(G|^*_{o^G[r',r]}, H|^*_{o^H[s',s]}) + w_{\lambda_G(\{r,r'\})}$
21: $maxval \leftarrow \max(maxval, val)$
22: **end if**
23: **end for**
24: **return** $maxval$
25: **else**
26: **return** 0
27: **end if**
28: **end function**

Algorithm 3 Computing $\sigma(G|^*_{o^G[v_b^G,v_e^G]}, H|^*_{o^H[v_b^H,v_e^H]})$

1: **procedure** MATCHBLOCK($G|^*_{o^G[v_b^G,v_e^G]}$, $H|^*_{o^H[v_b^H,v_e^H]}$)
2: Let T be the block-bridge tree of G and let b_G be the vertex of T corresponding to the common block B_G of v_b^G and v_e^G
3: Let U be the block-bridge tree of H and let b_H be the vertex of U corresponding to the common block B_H of v_b^H and v_e^H
4: $maxval \leftarrow 0$
5: **for all** vertices $v_m^G \in o^G[v_b^G, v_e^G] \setminus \{v_b^G, v_e^G\}$ and $v_m^H \in o^G[v_b^H, v_e^H] \setminus \{v_b^H, v_e^H\}$ **do**
6: $val \leftarrow 0$
7: $val \leftarrow val + \sigma(G|^*_{o^G[v_b^G,v_m^G]}, H|^*_{o^H[v_b^H,v_m^H]})$
8: **if** $\{v_b^G, v_e^G\} \in E(G)$ and $\{v_b^H, v_e^H\} \in E(H)$ and $\lambda_G(\{v_b^G, v_e^G\}) = \lambda_H(\{v_b^H, v_e^H\})$ **then**
9: **if** $val = 0$ **then**
10: $val \leftarrow \sigma(G|T_{b_G}^{v_b^G}, H|U_{b_H}^{v_b^H})$
11: **end if**
12: $val \leftarrow val + w_{\lambda_G(\{v_b^G,v_e^G\})}$
13: **end if**
14: **if** $val > 0$ and $\{v_m^G, v_e^G\} \in E(G)$ and $\{v_m^H, v_e^H\} \in E(H)$ and $\lambda_G(\{v_m^G, v_e^G\}) = \lambda_H(\{v_m^H, v_e^H\})$ **then**
15: $val \leftarrow val + w_{\lambda_G(\{v_m^G,v_e^G\})}$
16: $maxval \leftarrow \max(maxval, val)$
17: **end if**
18: **end for**
19: $\sigma(G|^*_{o^G[v_b^G,v_e^G]}, H|^*_{o^H[v_b^H,v_e^H]}) \leftarrow maxval$
20: **end procedure**

block-bridge tree (they have label #), then the size of the MCS between all the half-graphs with the children as end vertices will have to be considered during the matching process. At this point, these values will have been computed by the function MATCHBLOCK. Since we are using the BBP subgraph isomorphism, we do not consider matchings between regular vertices and vertices refering to blocks. If in the end a maximal matching is found, then there is a new solution for the non-block-splitting subgraph that was extended with one vertex.

6.1. Time complexity

In this section we will analyze the time complexity of our problem. In particular we will present the following theorem.

Theorem 1 *There is an algorithm computing the size of the maximal common subgraph under BBP subgraph isomorphism of two given graphs in time* $O(|V(G)^{7/2}.|V(H)|^{7/2})$.

Consider the algorithm as explained in the previous section. We start with some optimization notes. First, if MATCHEDGE is called multiple times with the same arguments we can avoid doing the computation again by remembering the result. Second, if two trees T_i^r and T_j^r are the same, we will consider only one of them.

A call to MATCHEDGE can be performed in unit time except if x and y refer to blocks. Line 19 is executed at most once for every ordered pair (r, r') and (s, s'), so for function MATCHEDGE we can account $O(|V(G)|.|V(H)|)$ time.

There are at most $O(|V(G)|.\Delta_G)$ elements in $\mathcal{T}(G)$ where Δ_G is the maximal degree of a ver-

tex of G. Therefore, MATCHVERTEX is called $O(|V(G)|.\Delta_G.|V(H)|.\Delta_H)$ times. Every call to MATCHVERTEX costs (except for the time spent in MATCHEDGE discussed above) at most the time to solve a weighted maximal matching problem, which can happen in cubic time (Munkres, 1957). Therefore, a rough bound on the time spent in MATCHVERTEX is $O(|V(G)|^{5/2}.\Delta_G.|V(H)|^{5/2}.\Delta_H)$.

In function MATCHBLOCK, the for loop is executed at most four times (once for every pair of orientations) for every value of v_b^G, v_m^G, v_e^G, v_b^H, v_m^H and v_e^H. Therefore, the time spent in MATCHBLOCK can be bounded by $O(|V(G)|^3.|V(H)|^3)$.

In summary, the above argument shows that there is an algorithm running in time bounded by a polynomial of degree 7. This is a quite rough bound which can be further improved by some more refined analysis which we omit due to space limitations.

7. Conclusions and further work

We have introduced a polynomial algorithm that computes the maximal common subgraph (MCS) of two outerplanar graphs under the block and bridge preserving subgraph isomorphism. We used this MCS notion to construct a metric on outerplanar graphs. It will be possible to use this metric as a similarity measure between molecules.

The next step is to conduct a series of experiments with the NCI (National Cancer Institute) database. This database contains thousands of molecules annotated with various activity levels. As a proof of concept, we will evaluate the performance of an instance-based learner that uses our metric. Next, we plan a full comparison with similar algorithms and metrics, in terms of efficiency as well as predictive performance.

Since 5.5% of the molecules in the NCI database cannot be represented by outerplanar graphs, we also plan to investigate other classes of graphs which would be suitable to represent molecules and for which we can design polynomial algorithms. One example is the class of graphs having a maximum treewidth. Treewidth expresses more or less the extent to which a graph resembles a tree.

Acknowledgements

Leander Schietgat is supported by a PhD grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen). Jan Ramon is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders. The authors thank Hendrik Blockeel for valuable suggestions and proofreading.

References

Akutsu, T. (1993). A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. *IEICE Transactions*, *E76-A*, 1488–1493.

Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, *18*, 689–694.

Bunke, H., & Shearer, K. (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, *19*, 255–259.

Diestel, R. (2000). *Graph theory*. Springer-Verlag.

Garey, M. R., & Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Freeman and Co.

Gillet, V. J., Wild, D. J., Willett, P., & Bradshaw, J. (1998). Similarity and dissimilarity methods for processing chemical structure databases. *The Computer Journal*, *41*, 547–558.

Horváth, T., Ramon, J., & Wrobel, S. (2006). Frequent subgraph mining in outerplanar graphs. *Proceedings of the Twelfth ACM SIGKDD*, 197–206.

McGregor, J. J. (1982). Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, *12*, 23–34.

Mitchell, S. L. (1979). Linear algorithms to recognize outerplanar and maximal outerplanar graphs. *Information Processing Letters*, *9*, 229–232.

Mitchell, T. (1997). *Machine learning*. McGraw-Hill.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, *5*, 32–38.

Raymond, J., Gardiner, E., & Willett, P. (2002). Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, *45*, 631–644.

Raymond, J., & Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, *16*, 521–533.

Syslo, M. (1982). The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, *17*, 91–97.