Finding Largest Common Substructures of Molecules in Quadratic Time

Andre Droschinsky, Nils Kriege(✉), and Petra Mutzel

Department of Computer Science, Technische Universität Dortmund,
Dortmund, Germany
{andre.droschinsky,nils.kriege,petra.mutzel}@tu-dortmund.de

Abstract. Finding the common structural features of two molecules is a fundamental task in cheminformatics. Most drugs are small molecules, which can naturally be interpreted as graphs. Hence, the task is formalized as maximum common subgraph problem. Albeit the vast majority of molecules yields outerplanar graphs this problem remains NP-hard.

We consider a variation of the problem of high practical relevance, where the rings of molecules must not be broken, i.e., the block and bridge structure of the input graphs must be retained by the common subgraph. We present an algorithm for finding a maximum common connected induced subgraph of two given outerplanar graphs subject to this constraint. Our approach runs in time $\mathcal{O}(\Delta n^2)$ in outerplanar graphs on n vertices with maximum degree Δ. This leads to a quadratic time complexity in molecular graphs, which have bounded degree. The experimental comparison on synthetic and real-world datasets shows that our approach is highly efficient in practice and outperforms comparable state-of-the-art algorithms.

1 Introduction

The maximum common subgraph problem arises in many application domains, where it is necessary to elucidate common structural features of objects represented as graphs. In cheminformatics this problem has been extensively studied [5,12,13] and is often referred to as maximum or *largest common substructure problem*. Two variants of the problem can be distinguished: The maximum common induced subgraph problem (MCIS) is to find isomorphic induced subgraphs of two given graphs with the largest possible number of vertices. The maximum common edge subgraph problem (MCES) does not require that common subgraphs are induced and aims at maximizing the number of edges. Both variants can be reduced to a maximum clique problem in the product graph of the two input graphs [12]. In cheminformatics MCES is used more frequently since it (i) reflects the notion of chemical similarity more adequately [12], and (ii) can reduce the running time of product graph based algorithms [11]. Although such algorithms still have exponential running time in the worst case, they are commonly applied to molecular graphs in practice [12].

This work was supported by the German Research Foundation (DFG), priority programme "Algorithms for Big Data" (SPP 1736).

B. Steffen et al. (Eds.): SOFSEM 2017, LNCS 10139, pp. 309–321, 2017.
DOI: 10.1007/978-3-319-51963-0_24

However, there are several restricted graph classes which render polynomial time algorithms possible [1,2,15]. The seminal work in this direction is attributed to J. Edmonds [10], who proposed a polynomial time algorithm for the maximum common subtree problem. Here, the given graphs and the desired common subgraph must be trees. Recently, it was shown that this problem can be solved in time $\mathcal{O}(\Delta n^2)$ for (unrooted) trees on n vertices with maximum degree Δ [3]. The (induced) subgraph isomorphism problem (SI) is to decide if a pattern graph is isomorphic to an (induced) subgraph of another graph and is generalized by MCIS and MCES, respectively. Both variants of SI are NP-complete, even when the pattern is a forest and the other graph a tree [6]; just as when the pattern is a tree and the other is outerplanar [14]. On the other hand, when both graphs are biconnected and outerplanar, induced SI can be solved in time $\mathcal{O}(n^2)$ [14] and SI in $\mathcal{O}(n^3)$ [9]. These complexity results and the demand in cheminformatics lead to the consideration of MCES under the so-called *block and bridge preserving* (BBP) constraint [13], which requires the common subgraph to retain the local connectivity of the input graphs. BBP-MCES is not only computable in polynomial-time, but also yields meaningful results for cheminformatics. A polynomial-time algorithm was recently proposed for BBP-MCIS, which requires time $\mathcal{O}(n^6)$ in series-parallel and $\mathcal{O}(n^5)$ in outerplanar graphs [7].

Most of the above mentioned polynomial time algorithms are either not applicable to molecular graphs or impractical due to high constants. A positive exception is the BBP-MCES approach of [13], which has been shown to outperform state-of-the-art algorithms on molecular graphs in practice. This algorithm is stated to have a running time of $\mathcal{O}(n^{2.5})$, but in fact leads to a running time of $\Omega(n^4)$ in the worst case [3].

Our Contribution. We take up the concept of BBP and propose a novel BBP-MCIS algorithm with running time $\mathcal{O}(\Delta n^2)$ in outerplanar graphs with n vertices and maximum degree Δ. We obtain this result by combining ideas of [3] for the maximum common subtree problem with a new algorithm for biconnected MCIS in biconnected outerplanar graphs. For this subproblem we develop a quadratic time algorithm, which exploits the fact that the outerplanar embedding of a biconnected outerplanar graph is unique. Moreover, the algorithm allows to list all solutions in quadratic total time. Our approach supports to solve BBP-MCIS w.r.t. a weight function on the mapped vertices and edges. The experiments show that BBP-MCIS in almost all cases yields the same results as BBP-MCES for molecular graphs under an adequate weight function. Our method outperforms in terms of efficiency the BBP-MCES approach of [13] by orders magnitude.

2 Preliminaries

We consider simple undirected graphs. Let $G = (V, E)$ be a graph, we refer to the set of *vertices* V by $V(G)$ or V_G and to the set of *edges* by $E(G)$ or E_G. An edge connecting two vertices $u, v \in V$ is denoted by uv or vu. The *order* $|G|$ of a graph G is its number of vertices. Let $V' \subseteq V$, then the graph $G[V'] = (V', E')$ with

$E' = \{uv \in E \mid u, v \in V'\}$ is called *induced* subgraph. For $U \subseteq V$ we write $G \backslash U$ for $G[V \backslash U]$. A graph is *connected* if there is a path between any two vertices. A *connected component* of a graph G is a maximal connected subgraph of G. A graph $G = (V, E)$ with $|V| \geq 3$ is called *biconnected* if $G \backslash \{v\}$ is connected for any $v \in V$. A maximal biconnected subgraph of a graph G is called *block*. If an edge uv is not contained in any block, the subgraph $(\{u, v\}, \{uv\})$ is called a *bridge*. A vertex v of G is called *cutvertex*, if $G \backslash \{v\}$ consists of more connected components than G. A graph is *planar* if it admits a drawing on the plane such that no two edges cross. The connected regions of the drawing enclosed by the edges are called *faces*, the unbounded region is referred to as *outer face*. An edge and a face are said to be *incident* if the edge touches the face. Two faces are *adjacent* if they are incident with a common edge. A graph is called *outerplanar* if it admits a drawing on the plane without crossings, in which every vertex lies on the boundary of the outer face. A *matching* in a graph $G = (V, E)$ is a set of edges $M \subseteq E$, such that no two edges share a vertex. A matching M is *maximal* if there is no other matching $M' \supsetneq M$ and *perfect*, if $2|M| = |V|$. A *weighted graph* is a graph endowed with a function $w : E \rightarrow \mathbb{R}$. A matching M in a weighted graph has weight by $W(M) := \sum_{e \in M} w(e)$; it is a *maximum weight matching* (MWM) if there is no matching M' of G with $W(M') > W(M)$.

An *isomorphism* between two graphs G and H is a bijection $\phi : V(G) \rightarrow V(H)$ such that $uv \in E(G) \Leftrightarrow \phi(u)\phi(v) \in E(H)$. A *common (induced) subgraph isomorphism* is an isomorphism between (induced) subgraphs $G' \subseteq G$ and $H' \subseteq H$. A subgraph $G' \subseteq G$ is *block and bridge preserving* (BBP) if (i) each bridge in G' is a bridge in G, (ii) any two edges in different blocks in G' are in different blocks in G. A common subgraph isomorphism ϕ is BBP if both subgraphs are BBP, it is *maximal* if it cannot be extended. Molecular graphs are typically annotated with atom and bond types, which should be preserved under isomorphisms. More general, we allow for a weight function $\omega : (V_G \times V_H) \cup (E_G \times E_H) \rightarrow \mathbb{R}^{\geq 0} \cup \{-\infty\}$. The weight $\mathcal{W}(\phi)$ of an isomorphism ϕ between G and H under ω is the sum of the weights $\omega(v, \phi(v))$ and $\omega(uv, \phi(v)\phi(v))$ for all vertices v and edges uv mapped by ϕ. A common subgraph isomorphism ϕ is *maximum* if its weight $\mathcal{W}(\phi)$ is maximum. A maximum isomorphism does not map any vertices or edges contributing weight $-\infty$ and we call these pairs *forbidden*. We further define $[1..k] := \{1, \ldots, k\}$ for $k \in \mathbb{N}$.

3 Biconnected MCIS in Outerplanar Graphs

In this section we present an algorithm to determine the weight of a maximum common biconnected induced subgraph isomorphism (2-MCIS) of two biconnected outerplanar graphs. First we show how to compute the maximal common biconnected subgraph isomorphisms. Since these may contain forbidden vertex and edge pairs, we then describe how to obtain the weight of a maximum solution from them. Finally we show how to output one or all maximum solutions.

Outerplanar graphs are well-studied and have several characteristic properties, see [14] for further information. In particular, our algorithm exploits the fact that biconnected outerplanar graphs have a unique outerplanar embedding in the plane (up to the mirror image). In these embeddings, every edge is incident to exactly two faces that are uniquely defined. We observe that the mapping is determined by starting parameters, i.e., an edge of both input graphs together with the mapping of their endpoints and incident faces.

We say a face is mapped by an isomorphism ϕ if all the vertices bordering the face are mapped by ϕ. We distinguish four cases to describe the mapping of an edge $uv \in E(G)$ to an edge $u'v' \in E(H)$ by an isomorphism ϕ between biconnected induced subgraphs. Assume the edge uv is incident to the faces A and B in G and $u'v'$ is incident to A' and B' in H, see Fig. 1(a). At least one face incident to uv must be mapped by ϕ, since the common subgraph must be biconnected. For the sake of simplicity of the case distinction, we also associate the two other faces, regardless of whether they are mapped or not. The isomorphism may map the endpoints of the edges in two different ways—just as the two incident faces. We can distinguish the following four cases: (1) $u \mapsto u',\ v \mapsto v',\ A \mapsto A',\ B \mapsto B'$, (2) $u \mapsto v',\ v \mapsto u',\ A \mapsto A',\ B \mapsto B'$, (3) $u \mapsto u',\ v \mapsto v',\ A \mapsto B',\ B \mapsto A'$, (4) $u \mapsto v',\ v \mapsto u',\ A \mapsto B',\ B \mapsto A'$.

Given an isomorphism ϕ between biconnected common induced subgraphs that maps the two endpoints of an edge e, let the function $\mathrm{type}(e, \phi) \in [1..4]$ determine the type of the mapping as above. The following result is the key to obtain our efficient algorithm.

Lemma 1. *Let ϕ and ϕ' be maximal isomorphisms between biconnected common induced subgraphs of the biconnected outerplanar graphs G and H. Assume $e \in E(G)$ is mapped to the same edge $e' \in E(H)$ by ϕ and ϕ', then*

$$\mathrm{type}(e, \phi) = \mathrm{type}(e, \phi') \Longleftrightarrow \phi' = \phi.$$

Proof. It is obvious that the direction $\Longleftarrow$ is correct. We prove the implication $\Longrightarrow$. Since the common subgraph is required to be biconnected, the isomorphisms ϕ and ϕ' both must map at least one face of G incident to the edge e to a face of H incident to e'. The two faces as well as the mapping of endpoints of the two edges are uniquely determined by the type of the mapping. We consider the mapping of the vertices on the cyclic border of these faces. Since the mapping of the endpoints of e are fixed, the mapping of all vertices on the border of the face is unambiguously determined. Since the common subgraph is required to be biconnected, every extension of the mapping must include all the vertices of a neighboring face. For this face, again, the mapping of the endpoints of the shared edge implicates the mapping of all vertices on the cyclic border and the extension is unambiguous. Therefore, the mapping can be successively extended to an unmapped face. Consequently $\phi(u) = \phi'(u)$ holds for all $u \in \mathrm{dom}(\phi) \cap \mathrm{dom}(\phi')$. Since ϕ and ϕ' are maximal it is not possible that one of them can be extended and, hence, we must have $\mathrm{dom}(\phi) = \mathrm{dom}(\phi')$ and the result follows. □

The proof of Lemma 1 constructively shows how to obtain a maximal solution given two edges $uv \in E(G)$, $u'v' \in E(H)$ and a type parameter $t \in [1..4]$. We assume that this approach is realized by the procedure MaximalIso$(uv, u'v', t)$, which returns the unique maximal isomorphism that maps the two given edges according to the specified type. The algorithm can be implemented by means of a tree structure that encodes the neighboring relation between inner faces, e.g., SP-trees as in [7,8] or weak dual graphs similar to the approach of [14]. The running time to compute a maximal solution ϕ then is $\mathcal{O}(|\phi|) \subseteq \mathcal{O}(n)$. Note that for some edge pairs not all four types of mappings are possible. The type $t \in [1..4]$ is *valid* for a pair of edges if at least one incident face can be mapped according to type t, i.e., the edges are incident to faces that are bordered by the same number of vertices.

A maximal solution ϕ may map vertex and edge pairs that are forbidden according to the weight function. In order to obtain the maximum weight, we split ϕ into *split isomorphisms* $\phi_1, \ldots, \phi_k$ such that each (i) has non-negative weight and (ii) again is an isomorphism between biconnected induced common subgraphs. The split isomorphisms can be obtained in time $\mathcal{O}(|\phi|)$ as follows. We consider the graph $G' = G[\mathrm{dom}(\phi)]$. For every forbidden edge uv that is incident to two inner faces in G', we split the graph into $G'_i[V(C_i) \cup \{u, v\}]$, where C_i is a connected component of $G' \setminus \{u, v\}$, $i \in [1..2]$. In these graphs we delete the forbidden vertices and edges and determine the blocks $B_1, \ldots, B_k$. Then ϕ, restricted to the vertices $V(B_i)$ of a block B_i, yields the split isomorphism ϕ_i for $i \in [1..k]$. This approach is realized by the function SplitIso(ϕ) used in the following. Every edge $e \in E(G)$ is mapped by at most one of the resulting isomorphisms, referred to by ϕ_e. Every 2-MCIS is a split isomorphism obtained from some maximal solution.

Algorithm 1 uses a table $D(e, f, t)$, $e \in E(G)$, $f \in E(H)$, $t \in [1..4]$ storing the weight of a 2-MCIS under the constraint that it maps e to f according to type t. The size of the table is $4|E(G)||E(H)| \in \mathcal{O}(nm)$, where $n = |V(G)|$ and $m = |V(H)|$. The algorithm starts with all pairs of edges and all valid types of mappings between them. For each, the maximal isomorphism between biconnected common induced subgraphs is computed by extending this initial mapping. By splitting the maximal solution, multiple valid isomorphisms with non-negative weight are obtained. These weights are then stored in D for all pairs of edges contained in ϕ considering the type of the mapping. This includes the $-\infty$ weights occurring if there are forbidden vertices or edges. Keeping these values allows to avoid generating the same isomorphism multiple times. The main procedure loops over all pairs of edges and the four possible mappings for each pair. Note that a mapping ϕ and its split isomorphisms are computed in time $\mathcal{O}(|\phi|) \subseteq \mathcal{O}(n)$. Improved analysis gives the following result.

Theorem 1. *Algorithm 1 computes the weight of a* 2*-MCIS between biconnected outerplanar graphs G and H in time $\mathcal{O}(|G||H|)$.*

Proof. We allocate the costs for a call of MaximalIso followed by SplitIso to cells of the table D. A mapping ϕ containing k edges is computed in time $\mathcal{O}(k)$ and as a result exactly k cells of the table D are filled with a value. The value of a cell is computed at most once: Line 2 assures that an edge mapping of a specific type is not used as initial mapping when the corresponding cell is

Algorithm 1. 2-MCIS in outerplanar graphs

```
  Input  : Biconnected outerplanar graphs $G$ and $H$.
  Output : Weight of a maximum common biconnected subgraph isomorphism.
  Data   : Table $D(e, f, t)$, $e \in E(G)$, $f \in E(H)$, $t \in [1..4]$ storing the weight of
           a 2-MCIS $\phi$ mapping $e$ to $f$ with $\text{type}(e, \phi) = t$.
1 forall the $uv \in E(G)$, $u'v' \in E(H)$ and $t \in [1..4]$ do
2     if type $t$ valid for $uv$ and $u'v'$ and $D(uv, u'v', t)$ undefined then
3         $\phi \leftarrow$ MaximalIso$(uv, u'v', t)$
4         $(\phi_1, \ldots, \phi_k) \leftarrow$ SplitIso$(\phi)$
5         forall the edges $e \in E(G)$ mapped to $f \in E(H)$ by $\phi$ do
6             $D(e, f, \text{type}(e, \phi)) \leftarrow \begin{cases} W(\phi_e) & \text{if } e \text{ is mapped by the split iso. } \phi_e \\ -\infty & \text{otherwise.} \end{cases}$
7 return maximum entry in $D$
```

already filled. Every initial mapping that is extended must lead to an isomorphism containing only edge mappings associated with undefined cells according to Lemma 1. Therefore the total costs of the algorithm can be allocated to cells of D, such that each cell pays a constant amount. This proves that the total running time is bounded by the size of the table, which is $\mathcal{O}(|G||H|)$. □

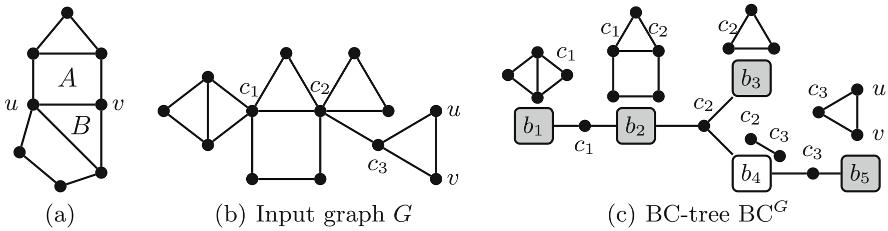
We can easily modify the algorithm to enumerate all maximum isomorphisms without affecting the total running time. First we run Algorithm 1 once to obtain the maximum weight $W_{\max}$. Then we run a modified version of Algorithm 1 that outputs every split isomorphism ϕ_i of size $W(\phi_i) = W_{\max}$ as soon as it is found, right after SplitIso(ϕ) is called in line 4.

4 Solving BBP-MCIS in Outerplanar Graphs

In the previous section we have presented an algorithm to compute a 2-MCIS between two biconnected outerplanar graphs. In this section we will generalize it to compute a BBP-MCIS between two outerplanar graphs G and H. In the following we assume the isomorphisms to be BBP. We require the input graphs to be connected. Otherwise we compute a BBP-MCIS for all pairs of connected components and select an isomorphism of maximum weight.

We proceed as follows. First, we give insight into the *BC-tree* data structure, which helps to partition the set $\mathcal{S}$ of all BBP common subgraph isomorphisms between G and H into subsets w.r.t. certain conditions. Then we compute an isomorphism of maximum weight in each of the subsets using a dynamic programming approach similar to the one used in [3] to solve the maximum common subtree problem. Among the computed isomorphisms we output one with maximum weight, thus a BBP-MCIS.

The BC-tree Data Structure. Given a BBP-MCIS, we can observe that bridges of G are mapped to bridges of H and that edges in one block of G can



(a) (b) Input graph G (c) BC-tree BC^G

Fig. 1. A biconnected outerplanar graph (a) with an edge uv incident to the faces A and B; a connected outerplanar graph (b) and its BC-tree (c). Block nodes have a gray background, while bridge nodes are not filled. The solid black nodes are the cutvertices. The corresponding subgraphs of G are shown above the block and bridge nodes.

only be mapped to edges contained in exactly one block of H, such that the mapped edges form a biconnected common subgraph. For a connected graph G let C^G denote the set of cutvertices, Bl^G the set of blocks and Br^G the set of bridges and $\mathrm{B}^G := \mathrm{Bl}^G \cup \mathrm{Br}^G$. The *BC-tree* BC^G of G is the tree with nodes $\mathrm{B}^G \cup \mathrm{C}^G$ and edges between nodes $b \in \mathrm{B}^G$ and $c \in \mathrm{C}^G$ iff $c \in V(b)$. We refer to the vertices of the BC-tree as B- and C-nodes and distinguish block nodes from bridge nodes. An example of a graph G and its BC-tree BC^G is shown in Fig. 1. For any graph G, we define $\mathrm{CC}(V', U)$ as the connected component of $G[V']$ that includes at least one vertex of U. We allow only such sets U, where the component is unambiguous. For example, in Fig. 1, $\mathrm{CC}(V_G \backslash V_{b_2}, V_{b_4})$ is the graph $G[\{c_3, u, v\}]$.

Partitioning of all BBP Isomorphisms $\boldsymbol{\mathcal{S}}$ into $\boldsymbol{\mathcal{S}} = \bigcup_x \boldsymbol{\mathcal{S}_x}$. First, we define $\mathcal{S}_1$ and $\mathcal{S}_2$. Let $b \in \mathrm{B}^G$ be an arbitrary block or bridge in G. We define $\mathcal{S}_1$ to contain all isomorphisms ϕ where at least one edge in b is mapped by the isomorphisms, i.e., $|\mathrm{dom}(\phi) \cap V(b)| \geq 2$. $\mathcal{S}_2$ is defined to contain all isomorphisms where exactly one vertex in b is mapped by the isomorphism. We can observe that S_1 and S_2 are disjoint and all other isomorphisms between G and H do not contain any vertices of b. Let $N = \{b_1, \ldots, b_k\} \subseteq \mathrm{B}^G$ be the blocks and bridges that share a cutvertex with b, i.e., $b_i \in N$ iff there is a node $c \in \mathrm{C}^G$ with bc and cb_i edges in the BC-tree BC^G. Any isomorphism ϕ that maps no vertex of b, maps vertices of at most one node b_i, because $G[\mathrm{dom}(\phi)]$ is connected by definition. For every b_i we recursively define sets $\mathcal{S}_x$ of isomorphisms as described above that map only vertices of $\mathrm{CC}(V_G \backslash V_b, V_{b_i})$.

As example consider Fig. 1(c) and let $b := b_2$. $\mathcal{S}_1$ consist of isomorphisms which map at least one edge of b_2 to an edge in H. The isomorphisms in $\mathcal{S}_2$ map exactly one vertex of $V(b)$ to H. The recursion continues on $N = \{b_1, b_3, b_4\}$. Three additional sets consist of isomorphisms which map at least one edge (and three more for exactly one vertex) of $V(b_i)$, $i \in \{1, 3, 4\}$, but no vertex of $V(b_2)$, operating on $\mathrm{CC}(V_G \backslash V_{b_2}, V_{b_i})$. The recursion for $b := b_4$ continues with $N = \{b_5\}$ and two additional sets. Some of the sets $\mathcal{S}_x$ are empty.

Partitioning of $\mathcal{S}_x$ into $\mathcal{S}_x = \bigcup_y \mathcal{P}_{xy}$. Before computing an isomorphism of maximum weight in a set $\mathcal{S}_x$, we partition $\mathcal{S}_x$ into subsets $\mathcal{P}_{x1}, \mathcal{P}_{x2}, \ldots$. The focus for the separation now is on the graph H. We distinguish two cases. If $\mathcal{S}_x$ is a set, where at least one edge of a certain block (bridge) b is mapped, then $\mathcal{S}_x$ is partitioned into $|\mathrm{Bl}^H|$ ($|\mathrm{Br}^H|$) subsets. The meaning is that for each B-node $\bar{b} \in Bl^H$ ($\bar{b} \in \mathrm{Br}^H$) the mapped vertices of the B-node $b \in \mathrm{B}^G$ are mapped only to $V(\bar{b})$. In terms of BBP this is block (bridge) preserving between b and $\bar{b}$, as intended. If $\mathcal{S}_x$ is a set, where exactly one vertex of b is mapped, the subsets are defined as follows. For each $(v, \bar{v}) \in V(b) \times V(H)$, where $\omega(v\bar{v}) \neq -\infty$ and v is in the CC we operate on, we define a subset with the restriction $\phi(v) = \bar{v}$.

Computing a Maximum Isomorphism in a Subset $\mathcal{P}_{xy}$. We now describe how to compute an isomorphism ϕ of maximum weight in a subset $\mathcal{P}_{xy} \subseteq \mathcal{S}_x$. The idea is to recursively extend mappings between some vertices of two single bridges or two single blocks along all pairs of mapped cutvertices into other B-nodes determined by MWMs, while preserving bridges and blocks. Between the computed isomorphisms we select one of maximum weight.

First, let $\mathcal{P}_{xy}$ be a subset, where at least one edge of a B-node $b \in \mathrm{B}^G$ has to be mapped to an edge of a B-node $\bar{b} \in B^H$. If b and $\bar{b}$ are bridges, the two possible mappings $V(b) \to V(\bar{b})$ are considered. If both are blocks, all maximal common biconnected subgraph isomorphisms between the blocks are considered (cf. Algorithm 1). We may have given a fixed mapping $v \mapsto \bar{v}$ (cf. (i) below). We call a considered isomorphism valid, if it respects the possible fixed mapping and contains only vertices of the CC we are operating on. We extend all the valid isomorphisms ϕ along all pairs $\phi(c) = \bar{c}, c \neq v$ of mapped cutvertices as follows. Let $B_c := \{b_1, \ldots b_k\}$, be the B-nodes of B^G, where bcb_i is a path, and $\bar{B}_c := \{\bar{b}_1, \ldots \bar{b}_l\}$, be the B-nodes of B^H, where $\bar{b}\bar{c}\bar{b}_j$ is a path, $i \in [1..k], j \in [1..l]$. For each pair $(b_i, \bar{b}_j) \in B_c \times \bar{B}_c$ we recursively calculate a BBP-MCIS φ_{ij} under the following restrictions: (i) The cutvertices must be mapped: $c \mapsto \bar{c}$. (ii) b_i and $\bar{b}_j$ are both bridges or both blocks. (iii) At least one other vertex in the block (bridge) b_i must be mapped, but only to $V(\bar{b}_j)$. Restriction (iii) assures that at least one vertex is added to the isomorphism. Therefore, the recursion to compute φ_{ij} is the method described in this paragraph. After computing φ_{ij} for each pair $(b_i, \bar{b}_j)$, we construct a weighted bipartite graph with vertices $B_c \uplus \bar{B}_c$ for each pair of mapped cutvertices. The weight of each edge $b_i\bar{b}_j$ is determined by the weight of a BBP-MCIS under the above restrictions, subtracted by $\omega(c, \bar{c})$ for the appropriate cutvertices c and $\bar{c}$. If there in no such restricted BBP-MCIS, there is no edge. Computing a MWM on each of the bipartite graphs determines the extension of ϕ. For each matching edge the corresponding computed isomorphisms are merged with ϕ. After extending all valid isomorphisms, we select one of maximum weight.

Second, let $\mathcal{P}_{xy}$ be a subset, where exactly one vertex v of $V(b)$ is mapped, and let $\phi(v) = \bar{v}$. If v is no cutvertex, the only possible expansion is within $V(b)$, which is not allowed in this subset. Therefore this subset contains exactly one isomorphism, $v \mapsto \bar{v}$. Next, assume v is a cutvertex. If $\bar{v}$ is a cutvertex, we may extend ϕ similar to the previous paragraph. In doing so, $c := v$, $\bar{c} := \bar{v}$ and B_c

as before. The only difference is $\bar{B}_c$, which is defined by all B-nodes containing $\bar{v} = \bar{c}$. The reason is that we have not mapped any other vertices yet, therefore we may expand in all directions in H. If $\bar{v}$ is no cutvertex, then $\bar{v}$ is contained in exactly one $\bar{b} \in B^H$. We are interested in BBP isomorphisms only. This means, all vertices that are mapped to $V(\bar{b})$ must be in the same block or bridge $b' \in \mathrm{B}^G$. Therefore, for each $b' \in \mathrm{B}^G$, where bvb' is a path and b' and $\bar{b}$ are of the same type (bridge/block), we compute an isomorphism with fixed mapping $v \mapsto \bar{v}$, where at least one edge of b' is mapped to $\bar{b}$. This falls back to the method of the above paragraph as well. Among the computed isomorphisms we select one of maximum weight. The pseudocode of the method described above is available in the extended version of this paper [4].

Time Complexity. The time to compute a BBP-MCIS essentially depends on the time to compute the BC-trees, the biconnected isomorphisms between the blocks of G and H, and the time to compute all the MWMs. The time to compute a BC-tree is linear in the number of edges and vertices. Considering all pairs of blocks and Theorem 1 we can bound the time for computing all the biconnected isomorphisms by $\mathcal{O}(\sum_b \sum_{\bar{b}} |V_b||V_{\bar{b}}|) \subseteq \mathcal{O}(|G||H|)$. We only need to compute MWMs for the pairs of cutvertices of the two graphs. It follows from the result of [3, Theorem 7] for the maximum common subtree problem, that the total time for this is $\mathcal{O}(|G||H|(\min\{\Delta^G, \Delta^H\} + \log\max\{\Delta^G, \Delta^H\}))$, where $\Delta^{\mathcal{G}}$ is the maximum degree of a C-node in $\mathrm{BC}^{\mathcal{G}}$. This proves the following theorem.

Theorem 2. *BBP-MCIS between two outerplanar graphs G and H can be solved in time $\mathcal{O}(|G||H|\Delta(G,H))$, where $\Delta(G,H) = 1$ iff G or H is biconnected or both are of bounded degree; otherwise $\Delta(G,H) = \min\{\Delta^G, \Delta^H\} + \log\max\{\Delta^G, \Delta^H\}$.*

5 Experimental Evaluation

In this section we evaluate our BBP-MCIS algorithm experimentally and compare to the BBP-MCES approach of [13].[1] Both algorithms were implemented in C++ and compiled with GCC v.4.8.4 as 64-bit application. Running times were measured on an Intel Core i7-3770 CPU using a single core. The available memory of 16 GB was sufficient for all the computations.

We are interested in answering the following questions:

(H1) To what extent differs BBP-MCIS from BBP-MCES on molecular graphs?
(H2) How large is the difference in terms of running time on molecular graphs?
(H3) How is the running time affected by specific properties of the input graphs?

To answer **(H1)** and **(H2)** we extracted 29000 randomly chosen pairs of outerplanar molecular graphs from a large chemical database.[2] The molecules in the database contain up to 104 vertices and 22 vertices on an average.

[1] We are grateful to Leander Schietgat for providing the implementation used in [13].
[2] NCI Open Database, GI50, http://cactus.nci.nih.gov.

The weight function ω was set to 1 for each pair of vertices and edges with the same label and $-\infty$ otherwise. This matches the setting in [13].

To answer **(H3)** we compared the algorithms on randomly generated connected outerplanar graphs. Our graph generator takes several parameters as input. With them we evaluated three different properties: the graph size, the average ratio $|E|/|V|$ of edges to vertices, and the average block size. For any outerplanar graphs the ratio of edges to vertices is less than 2. While evaluating the effect of one property, we preserved the other two. This procedure allows to verify whether our theoretical findings are consistent with the running times observed in practice. We set the weight function ω to 1 for each pair of vertices and edges, which corresponds to uniformly labeled graphs.

(H1). While comparing the weight of the isomorphisms computed by the two algorithms we observed a difference for only 0.40 % of the 29 000 tested molecule pairs. This suggests that BBP-MCIS yields a valid notion of similarity for outerplanar molecular graphs as it was shown for BBP-MCES [13].

(H2). Our algorithm computed the solutions on average 84 times faster. The dots in Fig. 2 represent the computation times of the two algorithms. The results are summarized in Table 1. Schietgat et al. [13] compared their BBP-MCES algorithm to a state-of-the-art algorithm for general MCIS. Their algorithm had

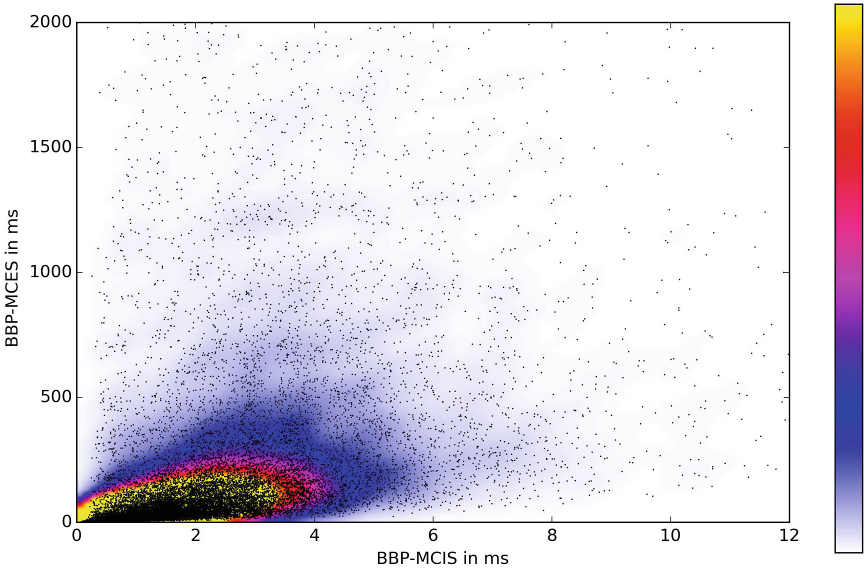


Fig. 2. Running times in ms for 28 399 BBP-MCIS computations. Each black dot represents a BBP-MCIS computation on two randomly chosen outerplanar molecular graphs. It directly compares the running time of our algorithm (MCIS, x-axis) and the implementation from [13] (MCES, y-axis). The running times of another 601 BBP-MCIS computations did not fit into the borders.

Table 1. Upper half: running times for our implementation (MCIS) and the implementation from [13] (MCES). Lower half: relative differences in computation times.

Algorithm	Average time	Median time	95 % less than	Maximum time
MCIS	1.97 ms	1.51 ms	5.28 ms	40.35 ms
MCES	207.08 ms	41.43 ms	871.48 ms	26 353.68 ms
Comparison	Average factor	Median factor	Minimum factor	Maximum factor
MCES/MCIS	83.8	25.6	1.8	28912.5

Table 2. Average time ± SD over 100 BBP-MCIS computations on random outerplanar graphs, varying one property (graph size, ratio of edges to vertices, block size BS). Note the units of measurement; timeout—total time exceeds 3 days.

Size	10	20	40	80	160
MCIS	0.7 ± 0.3 ms	2.3 ± 0.8 ms	8.2 ± 1.6 ms	33.5 ± 3.6 ms	133.2 ± 10.1 ms
MCES	207 ± 118 ms	3.4 ± 6.0 s	38.6 ± 90.6 s	234.2 ± 420.9 s	Timeout
$\|E\|/\|V\|$	0.98	1.10	1.24	1.46	1.78
MCIS	3.8 ± 0.3 ms	4.0 ± 1.1 ms	8.2 ± 1.6 ms	30.8 ± 4.0 ms	110.3 ± 11.6 ms
MCES	223 ± 16 ms	2.2 ± 2.6 s	38.6 ± 90.6 s	111.0 ± 213.8 s	216.1 ± 288.3 s
BS	3	5	10	20	40
MCIS	27 ± 6.4 ms	13.3 ± 2.4 ms	8.4 ± 1.7 ms	5.5 ± 1.4 ms	4.5 ± 0.9 ms
MCES	132 ± 14 ms	689 ± 548 ms	83.7 ± 118.7 s	30.4 ± 27.8 min	Timeout

similar computation times on small graphs and was much faster on large graphs. The maximum time of the general MCIS algorithm was more than 24 h. In contrast, our computation time never exceeded 41 ms. This clearly indicates that our algorithm is orders of magnitude faster than the general approach.

(H3). We first varied the size of the input graphs, while preserving an average ratio of edges to vertices of 1.24 and an average block size of 8. Based on Theorem 2 we expected the average time to increase by a factor of a bit more than 4, if we double the size. The results in Table 2 closely match this expectation.

Next, we evaluated different ratios of edges to vertices. The graph size was set to 40 and the average block size to 8. A higher ratio results in a higher number of faces in the blocks and consequently affects the time required by Algorithm 7. In particular, the table size and, thus, the running time is expected to show a quadratic growth. The increase in running time exceeds our expectation. This might be explained by the increasing size of the data structure used to represent the faces of the blocks.

Finally, we evaluated different average block sizes. The graph size was set to 40 and the average ratio of edges to vertices to 1.24. Higher block sizes mean less MWMs to compute, which are the most costly part in the BBP-MCIS computation. Therefore we expected the running time to decrease. The results shown in Table 2 support this.

6 Conclusion

We have developed an algorithm, which computes a well-defined, chemical meaningful largest common substructure of outerplanar molecular graphs in a fraction of a second. Hence, our method makes the graph-based comparison in large molecular datasets possible. As future work, we would like to extend our approach to more general graph classes with a focus on efficiency in practice.

References

1. Akutsu, T.: A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E76–A**(9), 1488–1493 (1993)
2. Akutsu, T., Tamura, T.: A polynomial-time algorithm for computing the maximum common connected edge subgraph of outerplanar graphs of bounded degree. Algorithms **6**(1), 119–135 (2013)
3. Droschinsky, A., Kriege, N., Mutzel, P.: Faster algorithms for the maximum common subtree isomorphism problem. MFCS 2016. LIPIcs, **58**, 34:1–34:14 (2016). arXiv:1602.07210
4. Droschinsky, A., Kriege, N., Mutzel, P.: Finding largest common substructures of molecules in quadratic time (2016). CoRR arXiv:1610.08739
5. Ehrlich, H.C., Rarey, M.: Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. Wiley Interdiscip. Rev. Comput. Mol. Sci. **1**(1), 68–79 (2011)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
7. Kriege, N., Kurpicz, F., Mutzel, P.: On maximum common subgraph problems in series-parallel graphs. In: Kratochvíl, J., Miller, M., Froncek, D. (eds.) IWOCA 2014. LNCS, vol. 8986, pp. 200–212. Springer, Heidelberg (2015). doi:10.1007/978-3-319-19315-1_18
8. Kriege, N., Mutzel, P.: Finding maximum common biconnected subgraphs in series-parallel graphs. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8635, pp. 505–516. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44465-8_43
9. Lingas, A.: Subgraph isomorphism for biconnected outerplanar graphs in cubic time. Theor. Comput. Sci. **63**(3), 295–302 (1989)
10. Matula, D.W.: Subtree isomorphism in $O(n^{5/2})$. In: Alspach, B., P.H., Miller, D. (eds.) Algorithmic Aspects of Combinatorics, Annals of Discrete Mathematics, vol. 2, pp. 91–106. Elsevier (1978)
11. Nicholson, V., Tsai, C.C., Johnson, M., Naim, M.: A subgraph isomorphism theorem for molecular graphs. In: Graph Theory and Topology in Chemistry. Studies in Physical and Theoretical Chemistry, no. 51, pp. 226–230. Elsevier (1987)
12. Raymond, J.W., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. J. Comput. Aided Mol. Des. **16**(7), 521–533 (2002)
13. Schietgat, L., Ramon, J., Bruynooghe, M.: A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics. Ann. Math. Artif. Intell. **69**(4), 343–376 (2013)

14. Sysło, M.M.: The subgraph isomorphism problem for outerplanar graphs. Theor. Comput. Sci. **17**(1), 91–97 (1982)
15. Yamaguchi, A., Aoki, K.F., Mamitsuka, H.: Finding the maximum common subgraph of a partial *k*-tree and a graph with a polynomially bounded number of spanning trees. Inf. Process. Lett. **92**(2), 57–63 (2004)