A Polynomial-Time Algorithm for Computing the Maximum Common Subgraph of Outerplanar Graphs of Bounded Degree

Tatsuya Akutsu and Takeyuki Tamura

Bioinformatics Center, Institute for Chemical Research, Kyoto University,
Gokasho, Uji, Kyoto 611-0011, Japan
{takutsu,tamura}@kuicr.kyoto-u.ac.jp

Abstract. This paper considers the maximum common subgraph problem, which is to find a connected graph with the maximum number of edges that is isomorphic to a subgraph of each of the two input graphs. This paper presents a dynamic programming algorithm for computing the maximum common subgraph of two outerplanar graphs whose maximum vertex degree is bounded by a constant, where it is known that the problem is NP-hard even for outerplanar graphs of unbounded degree. Although the algorithm repeatedly modifies input graphs, it is shown that the number of relevant subproblems is polynomially bounded and thus the algorithm works in polynomial time.

Keywords: maximum common subgraph, outerplanar graph, dynamic programming.

1 Introduction

Comparison of graph-structured data is important and fundamental in computer science. Among many graph comparison problems, the *maximum common subgraph problem* has applications in various areas, which include pattern recognition [4,14] and chemistry [12]. Although there exist several variants, the maximum common subgraph problem (MCS) usually means the problem of finding a connected graph with the maximum number of edges that is isomorphic to a subgraph of each of the two input undirected graphs.

Due to its importance in pattern recognition and chemistry, many practical algorithms have been developed for MCS and its variants [4,12,14]. Some exponential-time algorithms better than naive ones have also been developed [1,9]. Kann studied the approximability of MCS and related problems [10].

It is also important for MCS to study polynomially solvable subclasses of graphs. It is well-known that if input graphs are trees, MCS can be solved in polynomial time using maximum weight bipartite matching [6]. Akutsu showed that MCS can be solved in polynomial time if input graphs are almost trees of bounded degree whereas MCS remains NP-hard for almost trees of unbounded degree [2], where a graph is called almost tree if it is connected and the number

B. Rovan, V. Sassone, and P. Widmayer (Eds.): MFCS 2012, LNCS 7464, pp. 76–87, 2012.

of edges in each biconnected component is bounded by the number of vertices plus some constant. Yamaguchi et al. developed a polynomial-time algorithm for MCS and the maximum common induced connected subgraph problem for a degree bounded partial k-tree and a graph with a polynomially bounded number of spanning trees, where k is a constant [16]. However, the latter condition seems too strong. Schietgat et al. developed a polynomial-time algorithm for outerplanar graphs under the block-and-bridge preserving subgraph isomorphism [13]. However, they modified the definition of MCS by this restriction. Although it was announced that MCS can be solved in polynomial time if input graphs are partial k-trees and MCS must be k-connected (for example, see [3]), the restriction that subgraphs are k-connected is too strict from a practical viewpoint. On the subgraph isomorphism problem, which is closely related to MCS, polynomial-time algorithms have been developed for biconnected outerplanar graphs [11,15] and for partial k-trees with some constraints as well as their extensions [5,7].

In this paper, we present a polynomial-time algorithm for outerplanar graphs of bounded degree. Although this graph class is not a superset of the classes in previous studies [2,16], it covers a wide range of chemical compounds[1]. Furthermore, the algorithm or its analysis in this paper is not a simple extension or variant of that for the subgraph isomorphism problem for outerplanar graphs [11,15] or partial k-trees [5,7]. These algorithms heavily depend on the property that each connected component in a subgraph is not decomposed. However, to be discussed in Section 4, connected components from both input graphs can be decomposed in MCS and considering all decompositions easily leads to exponential-time algorithms. In order to cope with this difficulty, we introduce the concept of *blade*. The blade and its analysis play a key role in this paper.

2 Preliminaries

A graph is called *outerplanar* if it can be drawn on a plane so that all vertices lie on the outer face (i.e., the unbounded exterior region) without crossing of edges. Although there exist many embeddings (i.e., drawings on a plane) of an outerplanar graph, it is known that one embedding can be computed in linear time. Therefore, we assume in this paper that each graph is given with its planar embedding. A path is called *simple* if it does not pass the same vertex multiple times. In this paper, a path always means a simple path that is not a cycle.

A *cutvertex* of a connected graph is a vertex whose removal disconnects the graph. A graph is *biconnected* if it is connected and does not have a cutvertex. A maximal biconnected subgraph is called a *biconnected component*. A biconnected component is called a *block* if it consists of at least three vertices, otherwise it is an edge and called a *bridge*. An edge in a block is called an *outer* edge if it lies on the boundary of the outer face, otherwise called an *inner* edge. It is well-known that any block of an outerplanar graph has a unique Hamiltonian cycle, which consists of outer edges only.

[1] It was reported that 94.4% of chemical compounds in NCI database have outerplanar graph structures [8].

If we fix an arbitrary vertex of a graph G as the root r, we can define the parent-child relationship on biconnected components. For two vertices u and v, u is called *further* than v if every simple path from u to r contains v. A biconnected component C is called a *parent* of a biconnected component C' if C and C' share a vertex v, where v is uniquely determined, and every path from any vertex in C' to the root contains v. In such a case, C' is called a *child* of C. A cutvertex v is also called a *parent* of C if v is contained in both C and its parent component[2]. Furthermore, the root r is a parent of C if r is contained in C.

For each cutvertex v, $G(v)$ denotes the subgraph of G induced by v and the vertices further than v. For a pair of a cutvertex v and a biconnected component C containing v, $G(v, C)$ denotes the subgraph of G induced by vertices in C and its descendant components. For a biconnected component B with its parent cutvertex w, a pair of vertices v and v' in B is called a *cut pair* if $v \neq v'$, $v \neq w$, and $v' \neq w$ hold. For a pair (v, v') in B such that $v \neq v'$ holds (v or v' can be the parent cutvertex), $VB(v, v')$ denotes the set of the vertices lying on the one of the two paths connecting v and v' in the Hamilton cycle that does not contain the parent cutvertex except its endpoints. $B(v, v')$ is the subgraph of B induced by $VB(v, v')$ and is called a *half block*. It is to be noted that $B(v, v')$ contains both v and v'. Then, $G(v, v')$ denotes the subgraph of G induced by $VB(v, v')$ and the vertices in the biconnected components each of which is a descendant of some vertex in $VB(v, v') - \{v, v'\}$, and $\overline{G}(v, v')$ denotes the subgraph of G induced by the vertices in $G(v, v')$ and descendant components of v and v'.

Example. Fig. 1 shows an example of an outerplanar graph $G(V, E)$. Blocks and bridges are shown by gray regions and bold lines, respectively. B_1, B_3 and e_2 are the children of the root r. B_4, B_6 and B_7 are the children of B_3, whereas B_4 and B_6 are the children of w. Both w and B_3 are the parents of B_4 and B_6. $G(w)$ consists of B_4, B_5 and B_6, whereas $G(w, B_4)$ consists of B_4 and B_5. (v, v') is a cut pair of B_7, and $B_7(v, v')$ is a region surrounded by a dashed bold curve. $\overline{G}(v, v')$ consists of $B_7(v, v')$, B_8, B_9, B_{10}, e_4, e_5, and e_6, whereas $G(v, v')$ consists of $B_7(v, v')$, B_{10}, e_4, and e_5.

If a connected graph $G_c(V_c, E_c)$ is isomorphic to a subgraph of G_1 and a subgraph of G_2, we call G_c a *common subgraph* of G_1 and G_2. A common subgraph G_c is called a *maximum common subgraph* (MCS) of G_1 and G_2 if its number of edges is the maximum among all common subgraphs[3]. In this paper, we consider the following problem.

Maximum Common Subgraph of Outerplanar Graphs of Bounded Degree (OUTER-MCS)
Given two undirected connected outerplanar graphs G_1 and G_2 whose maximum vertex degree is bounded by a constant D, find a maximum common subgraph of G_1 and G_2.

[2] Both of a cutvertex and a biconnected component can be parents of the same component.

[3] We use MCS to denote both the problem and the maximum common subgraph.

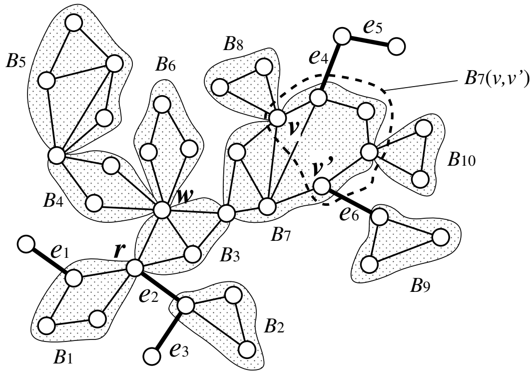

Fig. 1. Example of an outerplanar graph

Notice that the degree bound is essential because MCS is NP-hard for outerplanar graphs of unbounded degree even if each biconnected component consists of at most three vertices [2]. Although we do not consider labels on vertices or edges, our results can be extended to vertex-labeled and/or edge-labeled cases in which label information must be preserved in isomorphic mapping. In the following, n denotes the maximum number of vertices of two input graphs[4].

In this paper, we implicitly make extensive use of the following well-known fact [11] along with outerplanarity of input graphs.

Fact 1. *Let G_1 and G_2 be biconnected outerplanar graphs. Let $(u_1, u_2, \ldots, u_m)$ (resp. $(v_1, v_2, \ldots, v_n)$) be the vertices of G_1 (resp. G_2) arranged in the clockwise order in some planar embedding of G_1 (resp. G_2). If there is an isomorphic mapping $\{(u_1, v_{i_1}), (u_2, v_{i_2}), \ldots, (u_m, v_{i_m})\}$ from G_1 to a subgraph of G_2 then $v_{i_1}, v_{i_2}, \ldots, v_{i_m}$ appear in G_2 in either clockwise or counterclockwise order.*

3 Algorithm for a Restricted Case

In this section, we consider the following restricted variant of OUTER-MCS, which is called SIMPLE-OUTER-MCS, and present a polynomial-time algorithm for it: (i) any two vertices in different biconnected components in a maximum common subgraph G_c must not be mapped to vertices in the same biconnected component in G_1 (resp. G_2), (ii) each bridge in G_c must be mapped to a bridge in G_1 (resp. G_2), (iii) the maximum degree need not be bounded.

It is to be noted from the definition of a common subgraph (regardless of the above restrictions) that no two vertices in different biconnected components in G_1 (resp. G_2) are mapped to vertices in the same biconnected component in any common subgraph, or no bridge in G_1 (resp. G_2) is mapped to an edge in a block in any common subgraph.

[4] It should be noted that the number of vertices and the number of edges are in the same order since we only consider connected outerplanar graphs.

It seems that SIMPLE-OUTER-MCS is the same as one studied by Schietgat et al. [13]. Although our algorithm is more complex and less efficient than theirs, we present it here because the algorithm for a general (but bounded degree) case is rather involved and is based on our algorithm for SIMPLE-OUTER-MCS.

Here we present a recursive algorithm to compute the size of MCS in SIMPLE-OUTER-MCS, which can be easily transformed into a dynamic programming algorithm to compute an MCS. The following is the main procedure of the recursive algorithm.

Procedure $SimpleOuterMCS(G_1, G_2)$
$s_{\max} \leftarrow 0$;
for all pairs of vertices $(u, v) \in V_1 \times V_2$ **do**
  Let (u, v) be the root pair (r_1, r_2) of (G_1, G_2);
  $s_{\max} \leftarrow \max(s_{\max}, MCS_c(G_1(r_1), G_2(r_2)))$;
return $s_{\max}$.

The algorithm consists of recursive computation of the following three scores:

$MCS_c(G_1(u), G_2(v))$**:** the size of an MCS G_c between $G_1(u)$ and $G_2(v)$, where (u, v) is a pair of the roots or a pair of cutvertices, and G_c must contain a vertex corresponding to both u and v.

$MCS_b(G_1(u, C), G_2(v, D))$**:** the size of an MCS G_c between $G_1(u, C)$ and $G_2(v, D)$, where (C, D) is either a pair of blocks or a pair of bridges, u (resp. v) is the cutvertex belonging to both C (resp. D) and its parent, G_c must contain a vertex corresponding to both u and v, and G_c must contain a biconnected component (which can be empty) corresponding to a subgraph of C and a subgraph D.

$MCS_p(G_1(u, u'), G_2(v, v'))$**:** the size of an MCS G_c between $G_1(u, u')$ and $G_2(v, v')$, where (u, u') (resp. (v, v')) is a cut pair, and G_c must contain a cut pair (w, w') corresponding to both (u, u') and (v, v'). If there does not exist such G_c (which must be connected), its score is $-\infty$.

In the following, we describe how to compute these scores.

Computation of $MCS_c(G_1(u), G_2(v))$

As in the dynamic programming algorithm for MCS for trees or almost trees [2], we construct a bipartite graph and compute a maximum weight matching.

Let $C_1, \ldots, C_{h_1}, e_1, \ldots, e_{h_2}$ and $D_1, \ldots, D_{k_1}, f_1, \ldots, f_{k_2}$ be children of u and v respectively, where C_is and D_js are blocks and e_is and f_js are bridges (see Fig. 2). We construct an edge-weighted bipartite graph $BG(X, Y; E)$ by

$$X = \{C_1, \ldots, C_{h_1}, e_1, \ldots, e_{h_2}\}, \quad Y = \{D_1, \ldots, D_{k_1}, f_1, \ldots, f_{k_2}\},$$
$$E = \{(x, y) \mid x \in X, y \in Y\},$$
$$w(C_i, D_j) = MCS_b(G_1(u, C_i), G_2(v, D_j)), \quad w(C_i, f_j) = 0,$$
$$w(e_i, f_j) = MCS_b(G_1(u, e_i), G_2(v, f_j)), \quad w(e_i, D_j) = 0.$$

Then, we let $MCS_c(G_1(u), G_2(v))$ be the weight of the maximum weight bipartite matching of $BG(X, Y; E)$.

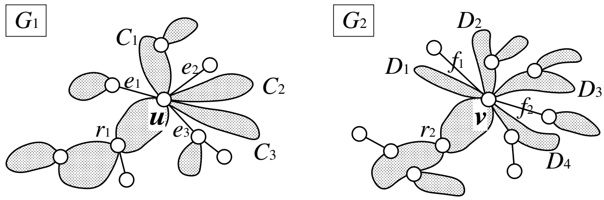

Fig. 2. Computation of $MCS_c(G_1(u), G_2(v))$

Computation of $MCS_b(G_1(u, C), G_2(v, D))$

Let $(u_1, u_2, \ldots, u_h)$ be the sequence of vertices in $G_1(u, C)$ such that there exists an edge $\{u_i, u\}$ for each u_i, where $u_1, u_2, \ldots, u_h$ are arranged in the clockwise order. $(v_1, v_2, \ldots, v_k)$ is defined for $G_2(v, D)$ in the same way. A pair of subsequences $((u_{i_1}, u_{i_2}, \ldots, u_{i_g}), (v_{j_1}, v_{j_2}, \ldots, v_{j_g}))$ is called an *alignment* if $i_1 < i_2 < \cdots < i_g$, and $j_1 < j_2 < \cdots < j_g$ or $j_g < j_{g-1} < \cdots < j_1$ hold[5] where $g = 0$ is allowed. We compute $MCS_b(G_1(u, C), G_2(v, D))$ by the following (see Fig. 3).

Procedure $SimpleOuterMCS_b(G_1(u, C), G_2(v, D))$
$s_{\max} \leftarrow 0$;
for all alignments $((u_{i_1}, u_{i_2}, \ldots, u_{i_g}), (v_{j_1}, v_{j_2}, \ldots, v_{j_g}))$ **do**;
  if C is a block and $g = 1$ **then continue**; /* blocks must be preserved */
  $s \leftarrow 0$;
  for $t = 1$ **to** g **do** $s \leftarrow s + 1 + MCS_c(G_1(u_{i_t}), G_2(v_{j_t}))$;
  for $t = 2$ **to** g **do** $s \leftarrow s + MCS_p(G_1(u_{i_{t-1}}, u_{i_t}), G_2(v_{j_{t-1}}, v_{j_t}))$;
  $s_{\max} \leftarrow \max(s, s_{\max})$;
return $s_{\max}$.

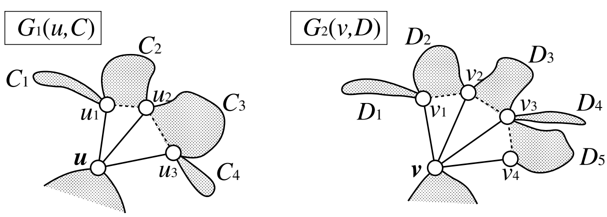

Fig. 3. Computation of $MCS_b(G_1(u, C), G_2(v, D))$

For example, consider an alignment $((u_1, u_2, u_3), (v_1, v_2, v_4))$ in Fig. 3, where all alignments are to be examined in the algorithm. Then, the score of this alignment is given by $3 + MCS_b(G_1(u_1, C_1), G_2(v_1, D_1)) + MCS_p(G_1(u_1, u_2),$

[5] The latter ordering is required for handling mirror images.

$G_2(v_1, v_2)) + MCS_p(G_1(u_2, u_3), G_2(v_2, v_4))$. In this case, an edge $\{v, v_3\}$ is removed and then v_3 is treated as a vertex on the path connecting v_2 and v_4 in the outer face.

Since the above procedure examines all possible alignments, it may take exponential time. However, we can modify it into a dynamic programming procedure as shown below, where we omit a subprocedure for handling mirror images. In this procedure, $u_1, u_2, \ldots, u_h$ and $v_1, v_2, \ldots, v_k$ are processed from left to right. In the first **for** loop, $M[s,t]$ stores the size of MCS between $G_1(u_s)$ and $G_2(v_t)$ plus one (corresponding to a common edge between $\{u, u_s\}$ and $\{v, v_t\}$). The second double **for** loop computes an optimal alignment. $M[s,t]$ stores the size of MCS between $G_1(u, C)$ and $G_2(v, D)$ up to u_s and v_t, respectively. $flag$ is introduced to ensure the connectedness of a common subgraph. For example, $flag = 0$ if $G_1(u)$ is a triangle but $G_2(v)$ is a rectangle.

for all $(s,t) \in \{1, \ldots, h\} \times \{1, \ldots, k\}$ **do**
  $M[s,t] \leftarrow 1 + MCS_c(G_1(u_s), G_2(v_t))$;
$flag \leftarrow 0$;
for $s = 2$ to h **do**
  for $t = 2$ to k **do**
    $M[s,t] \leftarrow M[s,t] +$
      $\max_{s'<s,t'<t}\{M[s',t'] + MCS_p(G_1(u_{s'}, u_s), G_2(u_{t'}, u_t))\}$;
    if $M[s,t] > -\infty$ **then** $flag \leftarrow 1$;
if C is a block and $flag = 0$ **then return** 0 **else return** $\max_{s,t} M[s,t]$.

Computation of $MCS_p(G_1(u,u'), G_2(v,v'))$

Let $(u_1, u_2, \ldots, u_h)$ be the sequence of vertices in $G_1(u, u')$ such that there exists an edge $\{u_i, u\}$ or $\{u_i, u'\}$ for each u_i, where $u_1, u_2, \ldots, u_h$ are arranged in the clockwise order. $(v_1, v_2, \ldots, v_k)$ is defined for $G_2(v, v')$ in the same way. For a pair (u_i, v_j), $l(u_i, v_j) = 1$ if $\{u_i, u\} \in E_1$ and $\{v_j, v\} \in E_2$ hold, otherwise $l(u_i, v_j) = 0$. For a pair (u_i, v_j), $r(u_i, v_j) = 1$ if $\{u_i, u'\} \in E_1$ and $\{v_j, v'\} \in E_2$ hold, otherwise $r(u_i, v_j) = 0$. We compute $MCS_p(G_1(u,u'), G_2(v,v'))$ by the following procedure, where it does not examine alignments with $j_g < j_{g-1} < \cdots < j_1$.

Procedure $SimpleOuterMCS_p(G_1(u,u'), G_2(v,v'))$
if $\{u,u'\} \in E_1$ and $\{v,v'\} \in E_2$ **then** $s_{\max} \leftarrow 1$ **else** $s_{\max} \leftarrow -\infty$;
for all alignments $((u_{i_1}, u_{i_2}, \ldots, u_{i_g}), (v_{j_1}, v_{j_2}, \ldots, v_{j_g}))$ **do**
  if $l(u_{i_t}, v_{j_t}) = 0$ and $r(u_{i_t}, v_{j_t}) = 0$ hold for some t **then continue**;
  if $l(u_{i_1}, v_{j_1}) = 0$ and $r(u_{i_g}, v_{j_g}) = 0$ hold **then continue**;
  if $\{u,u'\} \in E_1$ and $\{v,v'\} \in E_2$ **then** $s \leftarrow 1$ **else** $s \leftarrow 0$;
  for $t = 1$ **to** g **do** $s \leftarrow s + l(u_{i_t}, v_{j_t}) + r(u_{i_t}, v_{j_t}) + MCS_c(G_1(u_{i_t}), G_2(v_{j_t}))$;
  for $t = 2$ **to** g **do** $s \leftarrow s + MCS_p(G_1(u_{i_{t-1}}, u_{i_t}), G_2(v_{j_{t-1}}, v_{j_t}))$;
  $s_{\max} \leftarrow \max(s, s_{\max})$;
return $s_{\max}$.

This procedure returns $-\infty$ if there does not exist a connected common subgraph between $G_1(u,u')$ and $G_2(v,v')$ that contains (w,w') corresponding to both (u,u') and (v,v').

As an example, consider an alignment $((u_1, u_2, u_3, u_4), (v_1, v_2, v_3, v_5))$ in Fig. 4. Then, the score is given by $4+MCS_p(G_1(u_1, u_2), G_2(v_1, v_2))+MCS_p(G_1(u_2, u_3), G_2(v_2, v_3))+MCS_b(G_1(u_3, C_3), G_2(v_3, D_4))+MCS_p(G_1(u_3, u_4), G_2(v_3.v_5))$. For another example, the score is $-\infty$ for each of alignments $((u_1, u_3), (v_4, v_5))$, $((u_1, u_2), (v_1, v_2))$, and $((u_3), (v_3))$, whereas the score of $((u_2), (v_3))$ is 2.

As in the case of $SimpleOuterMCS_b(G_1(u, C), G_2(v, D))$, $SimpleOuter MCS_p$ $(G_1(u, u'), G_2(v, v'))$ can be modified into a dynamic programming version.

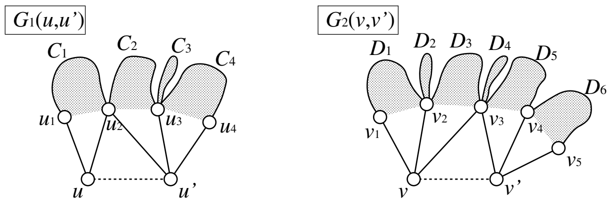

Fig. 4. Computation of $MCS_p(G_1(u, u'), G_2(v, v'))$

Then, we have the following theorem, where the proof is omitted here.

Theorem 1. *SIMPLE-OUTER-MCS can be solved in polynomial time.*

4 Algorithm for Outerplanar Graphs of Bounded Degree

In order to extend the algorithm in Section 3 for a general (but bounded degree) case, we need to consider decomposition of biconnected components. For example, consider graphs G_1 and G_2 in Fig. 5. We can see that in order to obtain a maximum common subgraph, biconnected components in G_1 and G_2 should be decomposed as shown in Fig. 5, where there are several other ways of optimal decompositions. This is the crucial point because considering all possible decompositions easily leads to exponential-time algorithms. In order to characterize decomposed components, we introduce the concept of *blade* as below.

Suppose that $v_{i_1}, \ldots, v_{i_k}$ are the vertices of a half block arranged in this order, and v_{i_1} and v_{i_k} are respectively connected to v and v', where v and v' can be

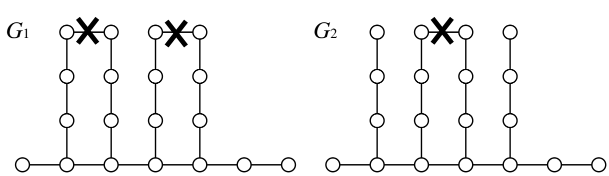

Fig. 5. Example of a difficult case

the same vertex. If we cut one edge $\{v_{i_h}, v_{i_{h+1}}\}$, we obtain two subgraphs, one induced by $v_{i_1}, v_{i_2}, \ldots, v_{i_h}$ and the other induced by $v_{i_k}, v_{i_{k-1}}, \ldots, v_{i_{h+1}}$, where only one such subgraph is obtained in the case of $i_1 = i_h$ or $i_k = i_{h+1}$, and no such subgraph is obtained in the case of $k = 2$. Each of these components is a chain of biconnected components called a *blade body*, and a subgraph consisting of a blade body and its descendants is called a *blade* (see Fig. 6). Vertices v_{i_1} and v_{i_k}, an edge $\{v_{i_h}, v_{i_{h+1}}\}$, and vertices $v_{i_h}, v_{i_{h+1}}$ are called *base vertices*, a *tip edge*, and *tip vertices*, respectively. The sequence of edges in the shortest path from v_{i_1} to v_{i_h} (resp. from v_{i_k} to $v_{i_{h+1}}$) is called the *backbone* of a blade. If $\{v, v_{i_1}\}$ is the leftmost edge (resp. $\{v', v_{i_k}\}$ is the rightmost edge) connecting to v (resp. v') and is removed, the resulting half block induced by $v_{i_k}, \ldots, v_{i_2}, v_{i_1}$ (resp. $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$) is also regarded as a blade body where v_{i_k} (resp. v_{i_1}) becomes the base vertex. For example, the rightmost blade in Fig. 7 is created by removing the rightmost edge of C_1.

Since a blade can be specified by a pair of base and tip vertices and an orientation (clockwise or counterclockwise), there exist $O(n^2)$ blades in G_1 and G_2. Of course, we need to consider the possibility that during the execution of the algorithm, other subgraphs may appear from which new blades are created. However, we will show later that blades appearing in the algorithm are restricted to be those in G_1 and G_2.

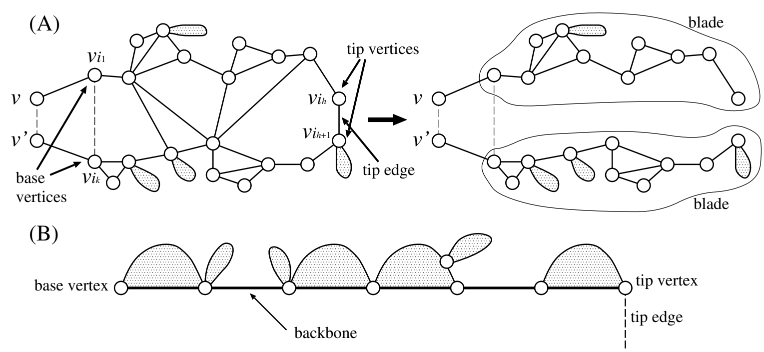

Fig. 6. (A) Construction of blades where subgraphs excluding gray regions (descendant components) are blade bodies, and (B) schematic illustration of a blade

4.1 Description of Algorithm

In this subsection, we describe the algorithm as a recursive procedure, which can be transformed into a dynamic programming one as in Section 3.

The main procedure ($OuterMCS(G_1, G_2)$) is the same as in Section 3, and we recursively compute three kinds of scores: $MCS_c(G_1(u), G_2(v))$, $MCS_b(G_1(u, C)$, $G_2(v, D))$, and $MCS_p(G_1(u, u'), G_2(v, v'))$, where cutvertices, cut pairs, blocks, and bridges do not necessarily mean those in the original graphs but may mean those in subgraphs generated by the algorithm.

Computation of $MCS_c(G_1(u), G_2(v))$

Let $C_1, \ldots, C_{h_1}$ and $e_1, \ldots, e_{h_2}$ be children of u, where C_is and e_js are blocks and bridges, respectively. Let $u_{i_1}, \ldots, u_{i_h}$ be the neighboring vertices of u that are contained in children of u. We define a *configuration* as a tuple of the following (see Fig. 7).

$s(u_{i_j}) \in \{0, 1\}$ for $j = 1, \ldots, k$: $s(u_{i_j}) = 1$ means that u_{i_j} is selected as a neighbor of u in a common subgraph, otherwise $s(u_{i_j}) = 0$. u_{i_j} is called a *selected vertex* if $s(u_{i_j}) = 1$.

$tip(u_{i_j}, u_{i_k})$: $e = tip(u_{i_j}, u_{i_k})$ is an edge in $B(u_{i_j}, u_{i_k})$ where B is the block containing u_{i_j}, u_{i_k}, and u. This edge is defined only for a consecutive selected vertex pair u_{i_j} and u_{i_k} in the same block (i.e., $B(u_{i_j}, u_{i_k})$ does not contain any other selected vertex). e is used as a tip edge where e can be empty which means that we do not cut any edge in $B(u_{i_j}, u_{i_k})$. It is to be noted that at most one edge in $B(u_{i_j}, u_{i_k})$ can be a tip edge and thus each $B(u_{i_j}, u_{i_k})$ is divided into at most two blade bodies: further decomposition will be done in later steps.

Each configuration defines a subgraph of $G_1(u)$ as follows.

- $e_i = \{u_{i_j}, u\}$ $(i \in \{1, \ldots, h_2\})$ remains if $s(u_{i_j}) = 1$. Otherwise e_i is removed along with its descendants.
- If no vertex in C_i is selected, C_i is removed along with its descendants. Otherwise, half blocks in C_i are broken into blade bodies (according to $s(\ldots)$s and $tip(\ldots)$s) and edges $\{u_{i_j}, u\}$ with $s(u_{i_j}) = 0$ are removed.

Let $C'_1, \ldots, C'_{p_1}$ and $e'_1, \ldots, e'_{p_2}$ be the resulting blocks and bridges containing u, which are new 'children' of u, for a configuration F_1. Configurations are defined for $G_2(v)$ in an analogous way. Let $D'_1, \ldots, D'_{q_1}$ and $f'_1, \ldots, f'_{q_2}$ be the resulting new children of v for a configuration F_2 of G_2. As in Section 3, we construct a bipartite graph BG_{F_1,F_2} by $w(C'_i, D'_j) = MCS_b(G_1(u, C'_i), G_2(v, D'_j))$, $w(C'_i, f'_j) = 0$, $w(e'_i, f'_j) = MCS_b(G_1(u, e'_i), G_2(v, f'_j))$, $w(e'_i, D'_j) = 0$, and compute the weight of the maximum weight matching for each configuration pair (F_1, F_2)[6]. The following is a procedure for computing $MCS_c(G_1(u), G_2(v))$.

Procedure $OuterMCS_c(G_1(u), G_2(v))$
$s_{\max} \leftarrow 0$;
for all configurations F_1 for $G_1(u)$ **do**
  for all configurations F_2 for $G_2(v)$ **do**
    $s \leftarrow$ weight of the maximum weight matching of BG_{F_1,F_2};
    if $s > s_{\max}$ **then** $s_{\max} \leftarrow s$;
return $s_{\max}$.

[6] Although a bridge cannot be mapped on a block here, a bridge can be mapped to an edge in a block by cutting the block using tip edge(s).

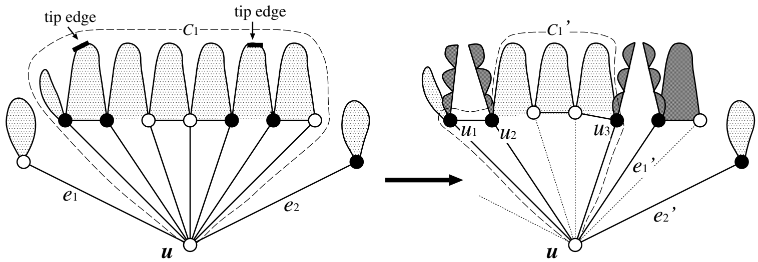

Fig. 7. Example of configuration and its resulting subgraph of $G_1(u)$, where black circles, dark gray regions, thin dotted lines denote selected vertices, blades, and removed edges, respectively. C'_1 has three blades and one block as the children, and e'_1 has two blades as the children. The role of u_1, u_2, and u_3 corresponds to that of u_1, u_2, and u_3 in Fig. 3.

Computation of $MCS_b(G_1(u, C'), G_2(v, D'))$

This score can be computed as in Section 3. In this case, we can directly examine all possible alignments because the number of neighbors of u or v is bounded by a constant and we need to examine a constant number of alignments.

Computation of $MCS_p(G_1(u, u'), G_2(v, v'))$**.**

This part is a bit more complex than the restricted case because we need to take configurations into account, where the details are omitted here.

4.2 Analysis

It is straightforward to check the correctness of the algorithm because it implicitly examines all possible common subgraphs. Therefore, we focus on analysis of the time complexity, where the proofs are omitted here. As mentioned before, each blade is specified by base and tip vertices in G_1 or G_2 and an orientation. Each half block is also specified by two vertices in a block in G_1 or G_2. We show that this property is maintained throughout the execution of the algorithm and bound the number of half blocks and blades as below.

Lemma 1. *The number of different half blocks and blades appearing in* $OuterMCS(G_1, G_2)$ *is* $O(n^2)$*.*

Finally, we obtain the following theorem.

Theorem 2. *A maximum connected common subgraph of two outerplanar graphs of bounded degree can be computed in polynomial time.*

5 Concluding Remarks

We have presented a polynomial-time algorithm for the maximum common subgraph problem for outerplanar graphs of bounded degree. However, it is not practically efficient. Therefore, development of a much faster algorithm is left as an open problem. Although the proposed algorithm might be modified for outputting all maximum common subgraphs, it would not be an output-polynomial time algorithm. Therefore, such an algorithm should also be developed.

References

1. Abu-Khzam, F.N., Samatova, N.F., Rizk, M.A., Langston, M.A.: The maximum common subgraph problem: faster solutions via vertex cover. In: Proc. 2007 IEEE/ACS Int. Conf. Computer Systems and Applications, pp. 367–373. IEEE (2007)
2. Akutsu, T.: A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. IEICE Trans. Fundamentals E76-A, 1488–1493 (1993)
3. Bachl, S., Brandenburg, F.-J., Gmach, D.: Computing and drawing isomorphic subgraphs. J. Graph Algorithms and Applications 8, 215–238 (2004)
4. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recognition and Artificial Intelligence 18, 265–298 (2004)
5. Dessmark, A., Lingas, A., Proskurowski, A.: Faster algorithms for subgraph isomorphism of k-connected partial k-trees. Algorithmica 27, 337–347 (2000)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, New York (1979)
7. Hajiaghayi, M., Nishimura, N.: Subgraph isomorphism, log-bounded fragmentation and graphs of (locally) bounded treewidth. J. Comput. Syst. Sci. 73, 755–768 (2007)
8. Horváth, T., Ramon, J., Wrobel, S.: Frequent subgraph mining in outerplanar graphs. In: Proc. 12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, pp. 197–206. ACM (2006)
9. Huang, X., Lai, J., Jennings, S.F.: Maximum common subgraph: some upper bound and lower bound results. BMC Bioinformatics 7(suppl. 4), S-4 (2006)
10. Kann, V.: On the Approximability of the Maximum Common Subgraph Problem. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 377–388. Springer, Heidelberg (1992)
11. Lingas, A.: Subgraph isomorphism for biconnected outerplanar graphs in cubic time. Theoret. Comput. Sci. 63, 295–302 (1989)
12. Raymond, J.W., Willett, P.: Maximum common subgraph isomorphism algorithms for the matching of chemical structures. J. Computer-Aided Molecular Design 16, 521–533 (2002)
13. Schietgat, L., Ramon, J., Bruynooghe, M.: A polynomial-time metric for outerplanar graphs. In: Proc. Workshop on Mining and Learning with Graphs (2007)
14. Shearer, K., Bunke, H., Venkatesh, S.: Video indexing and similarity retrieval by largest common subgraph detection using decision trees. Pattern Recognition 34, 1075–1091 (2001)
15. Syslo, M.M.: The subgraph isomorphism problem for outerplanar graphs. Theoret. Comput. Sci. 17, 91–97 (1982)
16. 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. Proc. Lett. 92, 57–63 (2004)