On Maximum Common Subgraph Problems in Series-Parallel Graphs*

Nils Kriege, Florian Kurpicz, and Petra Mutzel

Dept. of Computer Science, Technische Universität Dortmund, Germany
{nils.kriege,florian.kurpicz,petra.mutzel}@tu-dortmund.de

Abstract. The complexity of the maximum common connected subgraph problem in partial k-trees is still not fully understood. Polynomial-time solutions are known for degree-bounded outerplanar graphs, a subclass of the partial 2-trees. On the contrary, the problem is known to be **NP**-hard in vertex-labeled partial 11-trees of bounded degree. We consider series-parallel graphs, i.e., partial 2-trees. We show that the problem remains **NP**-hard in biconnected series-parallel graphs with all but one vertex of degree bounded by three. A positive complexity result is presented for a related problem of high practical relevance which asks for a maximum common connected subgraph that preserves blocks and bridges of the input graphs. We present a polynomial time algorithm for this problem in series-parallel graphs, which utilizes a combination of BC- and SP-tree data structures to decompose both graphs.

Keywords: maximum common subgraph, block and bridge preserving, series-parallel graphs

1 Introduction

Finding a maximum common subgraph (MCS) of two input graphs is an important task in many application domains like pattern recognition and cheminformatics [18]. MCS is well known to be **NP**-hard. Since practically relevant graphs, e.g., derived from small molecules, often have small treewidth [9], it is highly relevant to develop polynomial time algorithms for tractable graph classes and to clearly identify graph classes, where MCS remains **NP**-hard. For the related subgraph isomorphism problem such a clear demarcation for partial k-trees is known. Subgraph isomorphism is solvable in polynomial time in partial k-trees if the smaller graph either is k-connected or has bounded degree [13, 7]. However, it is **NP**-complete when the smaller graph is not k-connected or has more than k vertices of unbounded degree [8]. MCS apparently is at least as hard as subgraph isomorphism; two recent results show that it actually is considerably harder: Akutsu [2] has shown that finding a connected MCS is **NP**-hard in vertex-labeled partial 11-trees of bounded degree. Furthermore it was believed that the problem of finding a maximum common k-connected subgraph

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

of k-connected partial k-trees (k-MCS) can be solved with the same technique that was successfully used for subgraph isomorphism. Recently, it was shown that these techniques are insufficient even for series-parallel graphs, for which a new approach based on SP-trees was devised [10]. Further polynomial time algorithms were proposed for connected MCS of almost trees and outerplanar graphs of bounded degree [1, 3].

Motivated by the fact that even subgraph isomorphism is **NP**-hard when the smaller graph is a tree and the other is outerplanar, a problem variation referred to as BBP-MCS was considered [17, 18]. Here, the common subgraph is required to preserve blocks, i.e., maximal biconnected subgraphs, and bridges of the input graphs, which renders efficient algorithms for outerplanar graphs possible [17]. Notably, BBP-MCS yields meaningful results for molecular graphs in practice and even compares favorably to ordinary MCS in empirical studies [18, 16].

Our Contribution. On the theoretical side, we prove that finding a connected MCS of two biconnected series-parallel graphs with all but one vertex of degree bounded by three is **NP**-hard. We obtain this result by a polynomial-time reduction of the *Numerical Matching with Target Sums* problem. Furthermore, we consider BBP-MCS in series-parallel graphs and propose a polynomial time solution, thus, generalizing the known result for outerplanar graphs. Employing BC- and SP-tree decompositions of the input graphs allows us to identify subproblems closely related to k-MCS, $k \in \{1, 2\}$. We make use of a classical approach for the maximum common subtree problem [14], i.e., 1-MCS, and a recently proposed algorithm for 2-MCS [10] to obtain our main result. Our approach yields a running time of $\mathcal{O}(n^6)$ in series-parallel and $\mathcal{O}(n^5)$ in outerplanar graphs, where n is the maximum number of vertices in one of the input graphs.

2 Preliminaries

Let G be a simple graph. We denote the set of *vertices* by $V(G)$ and the set of *edges* by $E(G)$. A graph is *connected* if there is a path between any two vertices. Each maximal connected subgraph $G' \subseteq G$ is called *connected component*. Let $V \subseteq V(G)$, then $G[V]$ denotes the *induced subgraph* $G' \subseteq G$ with $V(G') = V$ and $E(G') = \{(u, v) \in V \times V : (u, v) \in E(G)\}$. A set $S \subseteq V(G)$ is called $|S|$*-separator* or *separator* of a connected graph G if $G \setminus S := G[V(G) \setminus S]$ consists of at least two connected components. If $S = \{v\}$ is a separator then v is called *cutvertex*. A graph G with $|V(G)| > k$ is called k*-connected* if there is no j-separator of G with $j < k$ and *biconnected* if $k = 2$. We define $[n] := \{1, \ldots, n\}$ for all $n \in \mathbb{N}$. A *path* is a sequence of vertices $(v_0, v_1, \ldots, v_n)$ such that $(v_{i-1}, v_i) \in E(G)$ for all $i \in [n]$. A path with $v_n = v_0$ is called *cycle*. The *length* of a path or cycle is the number of edges contained in it. Let $(v_0, v_1, \ldots, v_n)$ be a cycle, an edge (v_i, v_j) such that $1 \neq |i - j| < n$ is called *chord*. Cycles without chords are *chordless*.

A graph G is *bipartite* if there are two disjoint sets $U, U' \subseteq V(G)$ such that $U \cup U' = V(G)$ and for all $(u, v) \in E(G)$ neither $u, v \in U$ nor $u, v \in U'$. A *matching* of G is a set of edges $M \subseteq E(G)$ such that $u = u' \iff v = v'$ for all

$((u,v),(u',v')) \in M \times M$. The *maximum weighted bipartite matching* problem (MwbM) asks for the maximum weight of a matching of a weighted bipartite graph and is solvable in $\mathcal{O}(n^3)$, e.g., with the Hungarian method [11].

K_n denotes the complete graph with n vertices and $K_2^{s,t}$ denotes an instance of the K_2 where one vertex is called s- and the other t-vertex. A graph is *series-parallel* if each maximal biconnected subgraph can be constructed starting with a finite set of $K_2^{s,t}$ by performing a sequence of the following two operations.

S-Operation: Merge the s-vertex of one component with the t-vertex of a different component. The vertex created by merging remains unnamed.

P-Operation: Merge the s- and t-vertices of two different components of the set. The resulting vertices are called s- and t-vertex.

By definition, series-parallel graphs are at most biconnected and equivalent to partial 2-trees [4], i.e., graphs with treewidth at most 2. We use the notation and definition introduced in [5] to define the SP-tree decomposition of series-parallel graphs.

Definition 1 (SP-tree). *Let G be a biconnected series-parallel graph with at least three vertices. Then the SP-tree of G, denoted by* $\mathrm{SP}(G) = T^{\mathrm{SP}}$*, is the smallest tree such that the following conditions are satisfied:*

SP1 *each node*[1] *λ of T^{SP} is associated with a* skeleton *graph $S_\lambda = (V_\lambda, E_\lambda)$. Each edge $e = (u,v) \in E_\lambda$ is either a real or a virtual edge. If e is a virtual edge, then $S = \{u,v\}$ is a separator of G.*

SP2 *T^{SP} has two different types of nodes. S-nodes where the skeleton graph is a chordless cycle and P-nodes which have a skeleton graph consisting of multiple parallel edges between exactly two vertices.*

SP3 *for two adjacent nodes λ and η in T^{SP}, the skeleton graph S_λ contains a virtual edge e_η representing S_η and vice versa. The node η is called* pertinent *to the edge e_η.*

SP4 *The graph resulting by merging all skeleton graphs in a way that each virtual edge is replaced by the skeleton of its pertinent node in T^{SP} is exactly G.*

The sets of S-nodes and P-nodes in T^{SP} are denoted by $V_S(T^{\mathrm{SP}})$ or $V_P(T^{\mathrm{SP}})$ and T^{SP} is bipartite regarding these two sets. Let $r \in E(G)$, the *rooted SP-tree* is obtained by rooting T^{SP} at the node λ with $r \in V(S_\lambda)$. A rooted SP-tree induces a parent-child relation where a node λ is the parent of an adjacent node η if the path from the root node to λ is shorter than the path from the root node to η. If a node λ is the parent of a node η and $e_\lambda \in E(S_\eta)$ is the virtual edge pertinent to λ in η, then e_λ is called *reference edge* of λ and denoted by $\mathrm{ref}(\lambda)$.

Let G be a graph. Each maximal connected subgraph without a cutvertex with respect to that component is called a *block*. There are two different types of blocks: A maximal biconnected subgraph and a *bridge*, i.e., a K_2. Any two blocks of G may have at most one vertex in common, which must be a cutvertex. Blocks that are not bridges are called *non-bridge* block. Let B denote the set of blocks

[1] We call vertices of SP- and BC-trees nodes and vertices of the input graphs vertices.

of G and C the set of cutvertices of G. The graph with vertices $B \cup C$ and edges between each $b \in B$ and $c \in C$ iff $V(c) \in V(b)$ is called *block graph* of G and denoted by $\mathrm{BC}(G)$. If G is connected, the block graph is a tree and referred to as *BC-tree*. Each node Λ in a BC-tree has a skeleton graph S_Λ consisting of the vertices and edges represented by the node. Let $T^{\mathrm{BC}} = \mathrm{BC}(G)$ and $r \in E(G)$, the *rooted BC-tree* $T^{\mathrm{BC},r}$ is obtained by rooting T^{BC} at the B-node Λ such that $r \in V(S_\Lambda)$. It induces a parent-child relation as defined above in $T^{\mathrm{BC},r}$ and also a parent-child relation between the nodes of the SP-trees of the skeleton graphs. Since those only exists for non-bridge nodes, denoted by $V_{Bl}(T^{\mathrm{BC},r})$, there are two cases: Let T^{SP}_Λ be the SP-tree of the skeleton graph of $\Lambda \in V_B(T^{\mathrm{BC}})$. First, Λ is the root of T^{BC}, then T^{SP}_Λ is rooted at r. Otherwise, let Ξ be the parent of Λ hence Ξ is a cutvertex with $v = V(S_\Xi)$. Then T^{SP}_Λ is rooted at the P- or S-node such that v is in the skeleton graph of this node (P-node if existing). $V_{Br}(T^{\mathrm{BC},r})$ denotes the bridges of the BC-tree. Greek upper- and lowercase letters denote B-, C- and S-, P-nodes, resp. Latin letters denote vertices of the input graphs.

Let G and H be graphs. If $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then H is called *subgraph* of G. The graphs G and H are *isomorphic*, if there is a bijection $\phi\colon V(G) \to V(H)$, such that $(u,v) \in E(G) \iff (\phi(u),\phi(v)) \in E(H)\ \forall u,v \in V(G)$ and H is *subgraph isomorphic* to G, if H is isomorphic to a subgraph of G. We say u is *mapped* to v' if $\phi(u) = v'$. There is a *common subgraph isomorphism* between G and H, if there are sets $R \subseteq V(G)$ and $S \subseteq V(H)$ such that the induced subgraphs $G[R]$ and $H[S]$ are isomorphic. Let ϕ be the common subgraph isomorphism, then ϕ is a *maximum common subgraph isomorphism* if there is no common subgraph isomorphism ϕ' with $|\mathrm{dom}(\phi')| > |\mathrm{dom}(\phi)|$, where $\mathrm{dom}(\phi)$ denotes the *domain* of ϕ. A common subgraph is called *maximum common subgraph* (MCS) if there is no common subgraph containing more vertices.

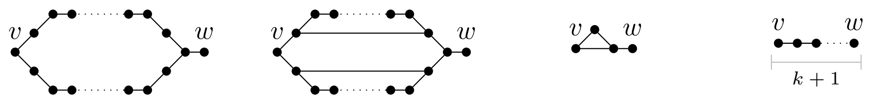
Definition 2 (Maximum Common Subgraph Problem (MCS)). *Given two graphs G and H, find the order of a maximum common connected subgraph.*

Please notice, that MCS can denote both: the problem and a subgraph. In the following we assume that the input graphs are connected series-parallel graphs and common subgraphs must be induced subgraphs of both input graphs.

3 MCS in Series-Parallel Graphs with Bounded Degree

In this section, we consider MCS where both input graphs are biconnected and have degree at most 3 for all but 1 vertex ($\mathrm{MCS}^{\leq 3,1}$). We prove that this problem is **NP**-hard and improve the result for subgraph isomorphism that, transferred to MCS, states that $\mathrm{MCS}^{\leq 4,2}$ is **NP**-hard [8].

Since the running time of an algorithm is given with respect to the size of the input, a reasonable encoding is demanded, e.g., an integer n can be encoded in $\log n$ bits. An **NP**-complete problem may no longer be **NP**-complete if the instances are encoded unary. *Strongly* **NP**-complete problems are **NP**-complete even if the input is encoded unary [6]. Hence even the values of numbers can be used. To prove that $\mathrm{MCS}^{\leq 3,1}$ is **NP**-hard we show that there is a polynomial-time reduction from the following problem which is strongly **NP**-complete [6].



(a) C_w^v (b) D_w^v (c) K_w^v (d) $P_w^{v,k}$

Fig. 1. Gadgets used to create the graphs G and H for a NMwTS instance.

Definition 3 (Numerical Matching with Target Sums (NMwTS)). *Given two disjoint sets X and Y with $|X| = |Y| = n$, a size function $s\colon X \cup Y \to \mathbb{Z}^+$ and a vector $\boldsymbol{b} = \langle b_1, b_2, \dots, b_n\rangle$ with $b_i \in \mathbb{Z}^+$ for all $i \in [n]$. Can $X \cup Y$ be partitioned into disjoint sets $A_1, A_2, \dots, A_n$ each containing one element from each of X and Y, such that $\sum_{a \in A_i} s(a) = b_i$ for all $i \in [n]$?*

3.1 Construction of the Polynomial-Time Reduction

For an instance $(X, Y, s, \boldsymbol{b})$ of NMwTS we construct two graphs, G and H to represent the values of the elements in X, Y and $\boldsymbol{b}$. Let $\varSigma_s := \sum_{z \in X \cup Y} s(z)$ and $\varSigma_{\boldsymbol{b}} := \sum_{i=1}^{n} b_i$. B_w^v denotes a cycle with $2\varSigma_s + 2$ vertices such that each path from v to w has length $\varSigma_s$. C_w^v is an instance of $B_{v'}^v$ with an additional vertex w called *anchor vertex* and an edge (v', w). D_w^v is an extension of C_w^v with two chords such that it is still outerplanar and there are two edge disjoint paths of length 4 from v to w.[2] K_w^v is an instance of K_3, where two vertices are denoted by v and v', with additional vertex w and an edge (v', w). Last, $P_w^{v,k}$ is a path of length k, where the vertices of degree 1 are denoted by v and w, see Fig. 1.

$$B_k = \bigcup_{i=1}^{n} \left(C_{c_i}^{\bar{x}} \cup D_{c_{i+n}}^{\bar{x}} \right) \cup \bigcup_{i=1+k}^{n+k-1} \left(P_{c_{i+1}}^{c_i,2} \right) \tag{1}$$

$$G = \bigcup_{i=1}^{n} \left(K_{\bar{x}_{1,i}}^{c_i} \cup K_{\bar{y}_{1,i}}^{c_{i+n}} \cup \bigcup_{j=2}^{s(x_i)} K_{\bar{x}_{j,i}}^{\bar{x}_{j-1,i}} \cup \bigcup_{j=2}^{s(y_i)} K_{\bar{y}_{j,i}}^{\bar{y}_{j-1,i}} \cup P_{\bar{y}_{s(y_i)}}^{\bar{x}_{s(x_i)},3} \right) \cup B_0 \tag{2}$$

$$H = \bigcup_{i=1}^{n} \left(K_{v_{1,i}}^{c_i} \cup K_{c_{i+n}}^{v_{b_i,i}} \cup \bigcup_{j=2}^{b_i} K_{v_j}^{v_{j-1}} \right) \cup B_n \tag{3}$$

The graphs G and H contain a subgraph which we call *base-gadget*, see Eq. 1. It consists of $2n$ cycles, $C_{c_i}^{\bar{x}}$ and $D_{c_{i+n}}^{\bar{x}}$ for $i \in [n]$. All cycles share the vertex $\bar{x}$, which is the only vertex with unbounded degree. The subgraphs $\bigcup_{i=1+k}^{n+k-1} P_{c_{i+1}}^{c_i,2}$, $k \in \{0, n\}$ are called *anchor paths* and are required to assure that G and H are biconnected. The index k in Eq. 1 is used to connect either the anchor vertices of cycles containing chords ($k = 0$) or of the chordless cycles ($k = n$). The graph G represents the values of the elements in X and Y, see Eq. 2. There is an *xy-path*

[2] If an instance of NMwTS does not allow the construction of D_w^v, all values are multiplied by 3.

between the anchor vertices c_i and c_{i+n} representing the values of x_i and y_i. The i-th xy-path consists of $s(x_i)$ connected K_w^v's (*x-path*) and $s(y_i)$ connected K_w^v's (*y-path*). The x- and the y-path are connected by one $P_w^{v,3}$ called *separating path.* Analogously, H represents the values in the vector $\boldsymbol{b}$, see Eq. 3. There is a *b-path* between the anchor vertices c_i and c_{i+n} representing the value $\boldsymbol{b}_i$. The i-th b-path consists of $b_i + 1$ K_w^v's.

Both, G and H, are series-parallel and can be computed in polynomial time with respect to the input size of NMwTS since the problem is **NP**-complete in the strong sense.

Lemma 1. *G and H are biconnected series-parallel graphs and can be constructed in polynomial time with respect to the values of the NMwTS instance.*

Proof (Sketch). Consider either G or H without $\bar{x}$, due to the anchor paths, the graph is connected, the same is true for G and H without any other vertex. Paths and cycles are series-parallel. Hence, K_w^v's are series-parallel and thus the xy-, b-paths and base-gadgets are series-parallel, too. They can be merged with P-operations such that $\bar{x}$ and an anchor vertex are the s- and t-nodes. Also G and H contain $|V(G)| = n\,(4\Sigma_s + 3) + 3\,(\Sigma_s + 1)\,, |V(H)| = n\,(4\Sigma_s + 3) + 3\,(\Sigma_{\boldsymbol{b}} + 1)\,, |E\,(G)\,| = 4\,(\Sigma_s n + \Sigma_s + n) + 3n - 2$ and $|E\,(H)\,| = 4\,(\Sigma_s n + \Sigma_{\boldsymbol{b}} + 2n) + n - 2$ vertices and edges, which is polynomial regarding the instance size of NMwTS. $\square$

Due to their construction, all MCS of G and H have common characteristics regarding their size and the vertices contained in them. First we show, that not all vertices in the xy- and b-paths can be contained in an MCS.

Lemma 2. *Let P be an xy-path and P' be a b-path each with an additional edge incident to the vertices with degree one, then an MCS of P and P' has size* $\min\left(|V(P)|, |V(P')|\right) - 1$.

Proof (Sketch). Due to their construction there are $k, l \in \mathbb{N}$ such that $3k = |V(P)|$ and $3l = |V(P')|$. If $k \leq l$, then the xy-path contains more than one K_3 less than the b-path. Since the separating path cannot be mapped to a K_3 there is at least one vertex which cannot be contained in an MCS. If $k > l$, then the xy-path contains at least two more K_3's than the b-path, hence each vertex except one in the b-path can be contained in the MCS. $\square$

We can also prove, that all vertices in the base gadgets are contained in the MCS except for the vertices only contained in the anchor paths.

Lemma 3. *Let B_0 and B_n be two base-gadgets, then an MCS of B_0 and B_n has size $|V(B_0)| - n + 1$.*

Proof (Sketch). The vertices with unbounded degree are mapped to each other, as otherwise not all cycles can be contained in the MCS. In B_0 the anchor paths are between the chordless cycles and in B_n the anchor paths are between the cycles with chords. If vertices of cycles of different types are mapped, then one vertex of each cycle and the adjacent anchor vertex cannot be contained in the MCS. Hence, only the $n - 1$ vertices contained in the anchor paths cannot be contained in the MCS. $\square$

3.2 Correctness of the Polynomial-Time Reduction

For the reduction, we show that an instance of NMwTS has a numerical matching if and only if an MCS of the corresponding graphs G and H has a specific size.

Lemma 4. *An instance $(X, Y, s, \boldsymbol{b})$ of NMwTS has a numerical matching if and only if $|V(G)| = |V(H)|$ and an MCS of G and H has size $|V(G)| - 2n + 1$.*

Proof (Sketch). Let $(X, Y, s, \boldsymbol{b})$ be an instance of NMwTS and G, H graphs constructed as described above. Assume that there is a numerical matching. Hence, $\Sigma_s = \Sigma_{\boldsymbol{b}}$ and thus $|V(G)| = |V(H)|$. An MCS of all xy-paths, b-paths and the base-gadgets has size $|V(G)| - n$ and $|V(G)| - n + 1$ (Lem. 2, 3). Even though they have been considered separately, the results can be combined, since all relevant vertices, the ones adjacent to the base-gadget or the xy-paths and b-paths, are contained in each MCS.

Now assume $|V(G)| = |V(H)|$ and there is an MCS with size $|V(G)| - 2n + 1$. Since we only consider connected MCSs, the vertex with unbounded degree must be contained in this MCS. For each xy-path and b-path there has to be one vertex which cannot be contained in an MCS (Lem. 2). The same is true for the base-gadgets, since the vertices of the anchor paths cannot be contained (Lem. 3). The vertices of the separating paths are not contained in an MCS. Thus the values of the elements of X and Y are correctly bipartitioned. Due to the size of the graphs for each b_i there is an x_j and $y_{j'}$ such that $b_i = s(x_j) + s(y_{j'})$. $\square$

Since G and H both have a maximum degree bounded by 3 for all but one vertex, the next result follows accordingly.

Theorem 1. MCS$^{\leq 3,1}$ *in biconnected series-parallel graphs is* ***NP****-hard.*

4 The Block-and-Bridge Preserving Maximum Common Subgraph Problem in Series-Parallel Graphs

In this section we consider the *block-and-bridge preserving* MCS (BBP-MCS) which has been introduced in [17] and also is used in [3]. An MCS is a BBP-MCS if it satisfies the following two conditions:

(BBP1) Any two vertices in different blocks in a common subgraph must not be contained in the same block of an input graph.

(BBP2) Each bridge in a common subgraph is a bridge in both input graphs.

We use the polynomial time algorithm for computing the size of a biconnected MCS of two biconnected series-parallel graphs (2-MCS) [10] to obtain an algorithm which solves BBP-MCS in arbitrary series-parallel graphs. To do so, we make use of a characteristic of an MCS: Every two vertices in an input graph which are not in the same block cannot be in the same block in any common subgraph. Hence, vertices in one block can only be mapped to vertices contained in exactly one block due to condition (BBP1). With respect to BBP-MCS, cutvertices have only to be considered if two of them are mapped. Since in biconnected graphs there are no cutvertices, BBP-MCS and 2-MCS are equivalent in those.

4.1 The Algorithm

We present an algorithm which solves BBP-MCS in polynomial time. The algorithm uses the BC-trees of the input graphs as underlying data structure. We apply the idea presented in [14] for MCS in trees, to the BC-trees. We decompose the BC-trees in rooted subtrees and compute the BBP-MCS for those. These results are then combined with MwbM. Since we want to solve BBP-MCS we do not need to compare all combination of subtrees, hence we use block split graphs to define the ones that must be considered. Let G be a series-parallel graph and $T_G^{\mathrm{BC},r}$ the BC-tree of G rooted at $r \in E(G)$. Let $S \subseteq V(G)$ be a 1- or 2-separator of G and $\{C_1, \dots, C_n\}$ the connected components of $G \setminus S$ such that $r \in E(C_1)$. Then $\overline{T_{G,S}^{\mathrm{BC},r}}$ denotes the induced subgraph $G[V(C_1) \cup S]$ and $T_{G,S}^{\mathrm{BC},r}$ denotes the induced subgraph $G[\bigcup_{i=2}^{n} C_i \cup S]$, called the *block split graphs* of G.

BBP-MCS is the main procedure of the algorithm. Given two series-parallel graphs it computes the size of a BBP-MCS. To do so, first the BC-trees and SP-trees of the non-bridge nodes are computed. Then the 2-MCS for each combination of bridges or non-bridge nodes is computed. For each of those combinations, the BC-trees are rooted at an edge r and r' in the skeleton graphs of these nodes. If the two nodes are non-bridge nodes, the BBP-MCS of those is computed using a 2-MCS algorithm modified to handle cutvertices, see Proc. 2.

BBP-MCS-S computes the 2-MCS of two non-bridge blocks [10, MCS-S]. It utilizes the rooted SP-trees given by the BC-tree decomposition. To obtain a BBP-MCS each common subgraph of a non-bridge block must be biconnected. Hence we traverse through the skeleton graph of the SP-trees and in the end

Algorithm 1 BBP-MCS(G, H)

```
Input: Two series-parallel graphs $G$ and $H$.
Output: Size of a BBP-MCS of the series-parallel graphs $G$ and $H$.
 1: $T_G^{\mathrm{BC}} \leftarrow \mathrm{BC}(G)$; $T_H^{\mathrm{BC}} \leftarrow \mathrm{BC}(H)$; $z \leftarrow 0$
 2: for all $(\Lambda, \Lambda') \in V_{\mathrm{Bl}}(T_G^{\mathrm{BC}}) \times V_{\mathrm{Bl}}(T_H^{\mathrm{BC}})$ do
 3:     $T_\Lambda^{\mathrm{SP}} \leftarrow \mathrm{SP}(S_\Lambda^{\mathrm{BC}})$; $T_{\Lambda'}^{\mathrm{SP}} \leftarrow \mathrm{SP}(S_{\Lambda'}^{\mathrm{BC}})$
 4:     for all $(\lambda, \lambda') \in V_S(T_\Lambda^{\mathrm{SP}}) \times V_S(T_{\Lambda'}^{\mathrm{SP}})$ do
 5:         $r \leftarrow$ arbitrary $(u, v) \in E(S_\lambda) \cap E(G)$; $\textsc{Root}(T_G^{\mathrm{BC}}, r)$
 6:         for all $r' = (u', v') \in E(S_{\lambda'}) \cap E(H)$ do
 7:             $\textsc{Root}(T_H^{\mathrm{BC}}, r')$
 8:             $p_1 \leftarrow$ BBP-MCS-S$(u, v, \lambda, u', v', \lambda')$
 9:             $p_2 \leftarrow$ BBP-MCS-S$(u, v, \lambda, v', u', \lambda')$
10:             $z \leftarrow \max(z, p_1, p_2)$
11: for all $(\Lambda, \Lambda') \in V_{\mathrm{Br}}(T_G^{\mathrm{BC}}) \times V_{\mathrm{Br}}(T_H^{\mathrm{BC}})$ do
12:     $r = (u, v) \leftarrow E(S_\Lambda^{\mathrm{BC}})$; $\textsc{Root}(T_G^{\mathrm{BC}}, r)$
13:     $r' = (u', v') \leftarrow E(S_{\Lambda'}^{\mathrm{BC}})$; $\textsc{Root}(T_H^{\mathrm{BC}}, r')$
14:     $p_1 \leftarrow$ BBP-MCS-C$(u, u')$ + BBP-MCS-C$(v, v')$
15:     $p_2 \leftarrow$ BBP-MCS-C$(u, v')$ + BBP-MCS-C$(v, u')$
16:     $z \leftarrow \max(z, p_1, p_2)$
17: return $z + 2$
```

Procedure 2 BBP-MCS-S$(u, v, \lambda, u', v', \lambda')$

```
Input: Vertices $u, v \in V(G), u', v' \in V(H)$ and $S$-nodes $\lambda \in V_S(T^{\mathrm{SP}}_{\cdot,G}), \lambda' \in V_S(T^{\mathrm{SP}}_{\cdot,H})$.
Output: Size of a BBP-MCS of $T^{\mathrm{BC},r}_{G,\{u,v\}}$ and $T^{\mathrm{BC},r}_{G,\{u',v'\}}$ such that $u \mapsto u'$ and $v \mapsto v'$.
 1: $e = (v, w) \leftarrow \textsc{Next}(v, \lambda)$; $e' = (v', w') \leftarrow \textsc{Next}(v', \lambda')$
 2: if $e = \mathrm{ref}(\lambda)$ then return BBP-MCS-S$(u, v, \mathrm{pS}(\lambda), u', v', \lambda')$
 3: if $e' = \mathrm{ref}(\lambda')$ then return BBP-MCS-S$(u, v, \lambda, u', v', \mathrm{pS}(\lambda'))$
 4: if $w = u$ and $w' = u'$ then return MCS-E$(e, \lambda, e', \lambda')$ + BBP-MCS-C$(w, w')$
 5: if $w = u$ xor $w' = u'$ then return $-\infty$
 6: $z \leftarrow$ MCS-E$(e, \lambda, e', \lambda')$+BBP-MCS-S$(u, w, \lambda, u', w', \lambda')$+BBP-MCS-C$(w, w')$+1
 7: if $e \notin E(G)$ or $e' \notin E(H)$ then
 8:     if $e \in E(G)$ then $M \leftarrow \{\lambda\}$ else $M \leftarrow \mathrm{cS}(e)$
 9:     if $e' \in E(H)$ then $M' \leftarrow \{\lambda'\}$ else $M' \leftarrow \mathrm{cS}(e')$
10:     for all $(\eta, \eta') \in M \times M'$ do
11:         $p \leftarrow$ BBP-MCS-S$(u, v, \eta, u', v', \eta')$
12:         $z \leftarrow \max(z, p)$
13: return $z$
```

have to return to the first visited vertex of the non-bridge block as otherwise the computed subgraph of the block is not biconnected. Whenever BBP-MCS-S is called, there are three cases regarding the edges incident to the considered vertices: If both are real, the extension of the mapping is straightforward. If both are virtual, the block split graphs $T^{\mathrm{BC},r}_{G,\{v,w\}}$ and $T^{\mathrm{BC},r'}_{H,\{v',w'\}}$ must be mapped, where v, w, v' and w' are the considered vertices. If one is real while the other is virtual, the real edges in an S-node pertinent to the virtual edge have to be considered. In addition to these cases, whenever two cutvertices w, w' are mapped, the block split graphs $T^{\mathrm{BC},r}_{G,\{w\}}$ and $T^{\mathrm{BC},r'}_{H,\{w'\}}$ must be mapped.

BBP-MCS-C computes the size of a BBP-MCS of two block split graphs obtained from cutvertices. Therefore, 0 is returned if the given vertices u and u' are not both cutvertices. Otherwise, we consider their child nodes $\mathrm{cB}(u)$ and $\mathrm{cB}(u')$ in the BC-trees rooted at r and r', respectively. To this end, we create a weighted complete bipartite graph C with vertex partition $\mathrm{cB}(u) \cup \mathrm{cB}(u')$. The weight $w\colon E(C) \to \mathbb{N} \cup \{-\infty\}$ of an edge is the size of a BBP-MCS of the two block split graphs associated with its endpoints. All edges incident to nodes associated with a block and a bridge have weight $-\infty$ as a mapping of those contradicts restriction (BBP1). It is important to notice, that the computation of the BBP-MCS of two blocks is not the same as in the main procedure, since the cutvertices must be mapped. Hence, we only consider mappings where these vertices are mapped, see Lines 8 and 9, Proc. 3. The child S-nodes of a P-node λ are denoted by $\mathrm{cS}(\lambda)$ and $\mathrm{pS}(\lambda)$ refers to its parent.

MCS-E is also called whenever the considered subgraph is extended by adding a new vertex to it. If both edges between the newly mapped vertex and the vertex added before are virtual, then the vertices are a separator, see (SP1) and the BBP-MCS of the block split graph has to be added to the result. $\textsc{Next}(u, \lambda)$ and $\textsc{Next}(u, \Lambda)$ return the vertex adjacent to u which yet has not

Procedure 3 BBP-MCS-C(u, u')

```
Input: Two cutvertices $u \in V(G)$, $u' \in V(H)$.
Output: Size of a BBP-MCS of $T^{\mathrm{BC},r}_{G,\{u\}}$ and $T^{\mathrm{BC},r}_{G,\{u'\}}$ such that $u \mapsto u'$.
 1: if $\nexists\lambda \in V_{\mathrm{C}}(T^{\mathrm{BC}}_G)\colon u \in V(S_\lambda)$ or $\nexists\lambda' \in V_{\mathrm{C}}(T^{\mathrm{BC}}_H)\colon u' \in V(S_\lambda)$ then return 0
 2: $M \leftarrow \mathrm{cB}(u)$; $M' \leftarrow \mathrm{cB}(u')$; $w \leftarrow \emptyset$
 3: for all $d = (\Lambda, \Lambda') \in V_{\mathrm{Bl}}(M) \times V_{\mathrm{Bl}}(M')$ do
 4:     $T^{\mathrm{SP}}_\Lambda \leftarrow \mathrm{SP}(S^{\mathrm{BC}}_\Lambda)$; $T^{\mathrm{SP}}_{\Lambda'} \leftarrow \mathrm{SP}(S^{\mathrm{BC}}_{\Lambda'})$
 5:     if $\exists\lambda \in V_P(T^{\mathrm{SP}}_\Lambda)\colon u \in V(S_\lambda)$ then $N \leftarrow \mathrm{cS}(\lambda)$ else $N \leftarrow \{\lambda\}$
 6:     if $\exists\lambda' \in V_P(T^{\mathrm{SP}}_{\Lambda'})\colon u' \in V(S_{\lambda'})$ then $N' \leftarrow \mathrm{cS}(\lambda')$ else $N' \leftarrow \{\lambda'\}$
 7:     for all $(\lambda, \lambda') \in N \times N'$ do
 8:         $(s, t) \leftarrow$ arbitrary $(s, t) \in E(S_\lambda) \cap E(G)\colon s = u$
 9:         for all $(s', t') \in E(S_{\lambda'}) \cap E(H)\colon s' = u'$ do
10:             $w(d) \leftarrow \max(w(d), \text{BBP-MCS-S}(s, t, \lambda, s', t', \lambda'))$
11: for all $d = (\Lambda, \Lambda') \in V_{\mathrm{Br}}(M) \times V_{\mathrm{Br}}(M)$ do
12:     $(u, v) \leftarrow \textsc{Next}(u, \Lambda)$; $(u', v') \leftarrow \textsc{Next}(u', \Lambda')$
13:     $w(d) \leftarrow \text{BBP-MCS-C}(v, v') + 1$
14: return $\textsc{MwbMatching}(M, M', w)$
```

Procedure 4 MCS-E$(e, \lambda, e', \lambda')$

```
Input: Edges $e = (u, v) \in E(S_\lambda)$ and $e' = (u', v') \in E(S'_\lambda)$
Output: Size of a BBP-MCS of $T^{\mathrm{BC},r}_{G,\{u,v\}}$ and $T^{\mathrm{BC},r}_{G,\{u',v'\}}$ such that $u \mapsto u'$ and $v \mapsto v'$.
 1: if $e \in E(G)$ xor $e' \in E(H)$ then return $-\infty$
 2: if $e \in E_r(S_\lambda)$ or $e' \in E_r(S_{\lambda'})$ then return 0
 3: $M \leftarrow \mathrm{cS}(e)$; $M' \leftarrow \mathrm{cS}(e')$; $w \leftarrow \emptyset$
 4: for all $d = (\eta, \eta') \in M \times M$ do
 5:     $w(d) \leftarrow \text{BBP-MCS-S}(u, v, \eta, u', v', \eta')$
 6: $p \leftarrow \textsc{MwbMatching}(M, M', w)$
 7: if $p = 0$, $e \notin E(G)$ or $e' \notin E(H)$ then return $-\infty$ else return $p$
```

been considered in the skeleton graph of λ and Λ, respectively. Root roots the BC-tree at the given edge and induces a rooting in all considered SP-trees. A more detailed description of the algorithm can be found in [12].

4.2 Analysis

We argue that Algorithm 1 solves BBP-MCS in polynomial time and show that if both input graphs are outerplanar then the running time can be improved.

Theorem 2. *Alg. 1 solves BBP-MCS in series-parallel graphs in time $\mathcal{O}(n^6)$.*

Proof. The correctness of the algorithm is based on the argumentation above and [10]. To prove the running time, we transform the algorithm in a dynamic programming approach. In [10, Th. 1] it is shown, that 2-MCS can be solved in time $\mathcal{O}(n^6)$ while storing the 2-MCS of two split graphs in a table of size $\mathcal{O}(n^4)$.

We assume w.l.o.g. that for each smaller block split graph the BBP-MCS has been computed whenever BBP-MCS-S is called. The size of BBP-MCS for each pair of child blocks has already been computed, hence the BBP-MCS of the block split graphs can be obtained with MwbM in $\mathcal{O}(n^3)$ (BBP-MCS-C). The tables have size $\mathcal{O}(n^4)$ and the only loop requires total time $\mathcal{O}(n^2)$ because the results have already been computed. Thus one call of BBP-MCS-S has running time $\mathcal{O}(n^4)$. Since there are only $\mathcal{O}(n^2)$ possible combinations of block split graphs the total running time of BBP-MCS-S is $\mathcal{O}(n^6)$.

BBP-MCS-C can be computed in time $\mathcal{O}(n^5)$ since the size of a BBP-MCS of the smaller block split graphs has already been computed. Again MwbM can be solved in time $\mathcal{O}(n^3)$ since at most $\mathcal{O}(n^2)$ of these matching problems must be solved, the total running time of BBP-MCS-C is $\mathrm{O}(n^5)$.

As MCS-E has not been changed with respect to [10], its running time is $\mathcal{O}(n^5)$, resulting in a total running time of $\mathcal{O}(n^6)$. □

Even though MwbM can be solved in $\mathcal{O}(n^3)$ it is a limiting factor regarding the running time. If we consider outerplanar graphs each P-node in the SP-trees has degree two which concludes in the following theorem.

Theorem 3. *BBP-MCS in outerplanar graphs can be solved in time* $\mathcal{O}\left(n^5\right)$.

Proof. The proof is similar to the proof of Theorem 2. Since all P-nodes in SP-trees of outerplanar graphs have degree two, the total running time of BBP-MCS-S reduces to $\mathcal{O}(n^4)$. Moreover, there is no need to use MwbM, as the bipartite graphs are K_2's. Consequently the running time of MCS-E is $\mathcal{O}(n^3)$.

BBP-MCS-C considers all adjacent nodes in the BC-tree whose number is not restricted if the graph is outerplanar and therefore still unbounded. Therefore the total running time is $\mathcal{O}(n^5)$. □

It was known that BBP-MCES in outerplanar graphs can be solved in $\mathcal{O}(n^7)$, where MCES refers to a variation of the problem that asks for edge-induced common subgraphs with maximum number of edges [17]. Note that — in contrast to the variant we consider — a subgraph where in one input graph two vertices are adjacent while the vertices in the other are not, is a feasible MCES.

5 Concluding Remarks

We have shown that MCS in series-parallel graphs with degree bounded by 3 for all but one vertex is **NP**-hard by reduction of NMwTS. Then we have extended a 2-MCS algorithm [10] to solve BBP-MCS with running time $\mathcal{O}(n^6)$. In outerplanar graphs, it can solve BBP-MCS in $\mathcal{O}(n^5)$ which is an improvement regarding the algorithm solving BBP-MCES in outerplanar graphs in $\mathcal{O}(n^7)$ [17]. BBP-MCES in outerplanar graphs was taken as basis to obtain polynomial time solutions for MCES in outerplanar graphs of bounded degree [3] and it has yet to be decided whether MCES in series-parallel graphs of bounded degree can be solved in polynomial time. To the author's best knowledge, there is only one problem which is known to be solvable in polynomial time in outerplanar graphs, but is **NP**-complete in series-parallel graphs: the *edge-disjoint paths problem* [15]. It still is unknown, whether MCS in series-parallel graphs is solvable in polynomial time if all vertices have bounded degree. Since the series-parallel graphs are equivalent to the partial 2-trees, there is a parameterized class of graphs, i.e., the partial k-trees, for which it is known that MCS is **NP**-complete for $k \geq 11$ even when the degree is bounded [2]. For all other $k > 1$, the complexity has yet to be decided.

References

1. Akutsu, T.: A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree. IEICE Trans. Fundamentals E76-A(9) (1993)
2. Akutsu, T., Tamura, T.: On the complexity of the maximum common subgraph problem for partial k-trees of bounded degree. In: Algorithms and Computation, pp. 146–155. Springer (2012)
3. 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)
4. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999)
5. Chimani, M., Hliněnỳ, P.: A tighter insertion-based approximation of the crossing number. In: Automata, Languages and Programming, pp. 122–134. Springer (2011)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman and Company, New York (1979)
7. Gupta, A., Nishimura, N.: Sequential and parallel algorithms for embedding problems on classes of partial k-trees. In: SWAT '94, LNCS, vol. 824 (1994)
8. Gupta, A., Nishimura, N.: The complexity of subgraph isomorphism for classes of partial k-trees. Theoretical Computer Science 164(1-2), 287–298 (1996)
9. Horváth, T., Ramon, J.: Efficient frequent connected subgraph mining in graphs of bounded tree-width. Theoretical Computer Science 411(31–33), 2784 – 2797 (2010)
10. Kriege, N., Mutzel, P.: Finding maximum common biconnected subgraphs in series-parallel graphs. In: MFCS. LNCS, vol. 8635, pp. 505–516. Springer (2014)
11. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly 2(1-2), 83–97 (1955)
12. Kurpicz, F.: Efficient Algorithms for the Maximum Common Subgraph Problem in Partial 2-Trees. Master's thesis, TU Dortmund (2014)
13. Matoušek, J., Thomas, R.: On the complexity of finding iso- and other morphisms for partial k-trees. Discrete Mathematics 108(1-3), 343–364 (1992)
14. Matula, D.W.: Subtree isomorphism in $O(n^{5/2})$. In: Algorithmic Aspects of Combinatorics, Ann. Discrete Math., vol. 2, pp. 91 – 106 (1978)
15. Nishizeki, T., Vygen, J., Zhou, X.: The edge-disjoint paths problem is np-complete for series–parallel graphs. Discrete Applied Mathematics 115(1), 177–186 (2001)
16. Schietgat, L., Costa, F., Ramon, J., De Raedt, L.: Effective feature construction by maximum common subgraph sampling. Machine Learning 83(2), 137–161 (2011)
17. Schietgat, L., Ramon, J., Bruynooghe, M.: A polynomial-time metric for outerplanar graphs. In: Mining and Learning with Graphs (MLG) (2007)
18. Schietgat, L., Ramon, J., Bruynooghe, M., Blockeel, H.: An efficiently computable graph-based metric for the classification of small molecules. In: Discovery Science, LNCS, vol. 5255, pp. 197–209. Springer Berlin / Heidelberg (2008)