On Maximum Common Subgraph Problems in Series-Parallel Graphs⋆

Nils Kriege, Florian Kurpicz, Petra Mutzel

Technische Universität Dortmund, Department of Computer Science
Chair of Algorithm Engineering (LS11), 44221 Dortmund, Germany

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 other hand, 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 3. 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 of two input graphs is an important task in many application domains like pattern recognition and cheminformatics [1]. The associated problem (MCS) is well known to be **NP**-hard. Since practically relevant graphs, e.g., derived from small molecules, often have small treewidth [2], 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 [3, 4]. However, it is **NP**-complete when the smaller graph is not k-connected or has more than k vertices of unbounded degree [5]. MCS is at least as hard as subgraph isomorphism; two recent results show that it actually is considerably harder: Akutsu [6] 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 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 [7]. Further polynomial time algorithms were proposed for connected MCS of almost trees and outerplanar graphs of bounded degree [8, 9].

Motivated by the fact that even subgraph isomorphism is **NP**-hard when the smaller graph is a tree and the other is outerplanar [10], a problem variation referred to as BBP-MCS (block-and-bridge preserving MCS) was considered [11, 1, 12]. 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 [11]. Notably, BBP-MCS yields meaningful results for molecular graphs in practice and even compares favorably to ordinary MCS solution algorithms in empirical studies [1, 13].

Our Contribution. On the theoretical side, we prove that finding a connected MCS of two biconnected series-parallel graphs, i.e., graphs with treewidth at most 2 [14], 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 [15], i.e., 1-MCS, and a recently proposed algorithm for 2-MCS [7] to obtain our 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

We consider *simple* graphs, i.e., a graph G without loops and multiple edges. We denote the finite set of *vertices* by $V(G)$ and the finite set of *edges* by $E(G)$. A graph G' is a *subgraph* of G, denoted by $G' \subseteq G$, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. We say a subgraph $G' \subseteq G$ is *maximal* regarding a property P if there is neither a vertex $v \in V(G)$ nor an edge $e \in E(G)$ such that $G' \cup \{v\} \cup \{e\}$ still satisfies P, where $G' \cup \{v\} \cup \{e\}$ is a graph with edges $E(G') \cup \{v\}$ and vertices $V(G') \cup \{e\}$. A graph is *connected* if there is a path between any two vertices.
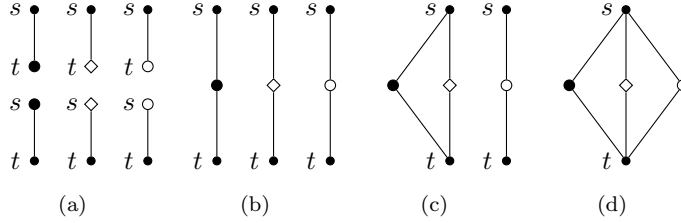
Figure 1: (a) A set of six $K_2^{s,t}$, (b) the set after an S-operation is applied to the s- and t-nodes of the same shape, (c) the set after a P-operation is applied to the first two components and finally (d) the set after a P-operation is applied to the remaining pair of s- and t-nodes resulting in a series-parallel graph.

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 separator S is said to *separate* two vertices $a, b \in V(G)$ if a and b are in different connected components of $G \setminus S$. A separator S of G is called *minimal* if there are vertices $a, b \in V(G)$ that are separated by S and there is no separator $S' \subset S$ that separates a and b. A graph G with $|V(G)| > k$ is called k-*connected* if there is no j-separator of G such that $j < k$ and *biconnected* if $k = 2$. We define $[n] := \{1, \ldots, n\}$ for all $n \in \mathbb{N}$. A sequence of distinct vertices $(v_0, v_1, \ldots, v_n)$ such that $(v_{i-1}, v_i) \in E(G)$ for all $i \in [n]$ is called *path*. If all but the first and the last vertex are distinct, i.e., $v_n = v_0$, the sequence is called *cycle*. The *length* of a path or cycle is the number of edges contained in it. Let $(v_n, v_1, \ldots, v_n)$ be a cycle, an edge (v_i, v_j) such that $1 \neq |i - j| < n$ is called *chord*. Cycles without chords are called *chordless* cycles.

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* in G is a set of edges $M \subseteq E(G)$ such that $u = u' \iff v = v'$ for all $((u,v),(u',v')) \in M \times M$. In a *weighted* graph G, each edge $e \in E(G)$ is associated with a weight $w(e) \in \mathbb{R} \cup \{-\infty\}$. The *maximum weighted bipartite matching* problem (MwbM) asks for the maximum weight of a matching in a weighted bipartite graph and is solvable in $\mathcal{O}(n^3)$, e.g., with the Hungarian method [16].

A graph is *complete* if there is an edge between any two vertices. 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, e.g. Figure 1.

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

3

P-**Operation** Given two different connected components of the set, merge both s-vertices with each other and both t-vertices with each other. The resulting vertices are called s- and t-vertex, respectively.

By definition, series-parallel graphs are at most biconnected and equivalent to partial 2-trees [14], i.e., graphs with treewidth at most 2.

The graphs G and H are said to be *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. The problem we consider is defined as follows.

Definition 1 (Maximum Common Subgraph Problem (MCS)). Given two input graphs G and H, return the number of vertices in a maximum common connected induced 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 connected 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 one 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 [5].

Since the running time of an algorithm is given with respect to the size of the input, a reasonable encoding is demanded, e.g., the 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 [17]. Hence, even the values of numbers can be used in a polynomial-time reduction. 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 **NP**-complete in the strong sense [18].

Definition 2 (Numerical Matching with Target Sums (NMwTS)).
Given two disjoint sets X and Y of equal size $|X| = |Y| = n$, a value function $s \colon X \cup Y \to \mathbb{Z}^+$ associating each element in X and Y with positive integer value and a vector $\vec{b} = \langle b_1, b_2, \ldots, b_n \rangle$ with $b_i \in \mathbb{Z}^+$ for all $i \in [n]$. NMwTS asks whether $X \cup Y$ can be partitioned into disjoint sets $A_1, A_2, \ldots, A_n$ such that each
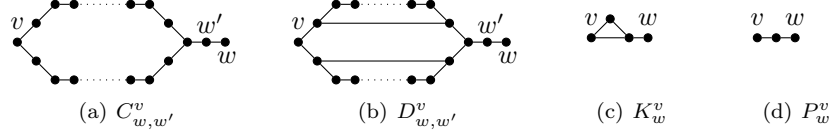
(a) $C^v_{w,w'}$ (b) $D^v_{w,w'}$ (c) K^v_w (d) P^v_w

Figure 2: Types of gadgets used to construct the graphs G and H.

set contains one element from X and one element from Y with $\sum_{a \in A_i} s(a) = b_i$ for all $i \in [n]$?

3.1. Construction of the Polynomial-Time Reduction

For an instance $(X, Y, s, \vec{b})$ of NMwTS we construct two graphs, which represent the values of the elements in X, Y (graph G) and $\vec{b}$ (graph H) in a way such that the number of vertices in an MCS of G and H indicates whether there is a numerical matching in the NMwTS instance. Let $\Sigma_s := \sum_{z \in X \cup Y} s(z)$ and $\Sigma_{\vec{b}} := \sum_{i=1}^n b_i$. First we define some gadgets which are then used to construct the graphs. The gadget B^v_w denotes a cycle with $2\Sigma_s + 2$ vertices such that both paths from v to w have length Σ_s. Next, the gadget $C^v_{w,w'}$ is a $B^v_{v'}$ with additional vertices w, w' called *anchor vertex* and *prime anchor vertex*, resp., and additional edges $(v', w'), (w', w)$, see Figure 2(a). The gadget $D^v_{w,w'}$ is an extension of $C^v_{w,w'}$ with two chords such that it is still outerplanar and there are two edge disjoint paths of length 4 from v to w, see Figure 2(b).[1] Lats, the gadget K^v_w is a K_3, where two vertices are denoted by v and v', with an additional vertex w and an edge (v', w), see Figure 2(c). Last, P^v_w is a path of length 2, where the vertices of degree 1 are denoted by v and w, see Figure 2(d). We construct the graphs G and H as follows.

$$B_G = \bigcup_{i=1}^n \left(C^{\bar{x}}_{c_i, c'_i} \cup D^{\bar{x}}_{c_{i+n}, c'_{i+n}} \right) \cup \bigcup_{i=1}^{n-1} \left(P^{c'_i}_{c'_{i+1}} \right) \tag{1}$$

$$B_H = \bigcup_{i=1}^n \left(C^{\bar{y}}_{c_i, c'_i} \cup D^{\bar{y}}_{c_{i+n}, c'_{i+n}} \right) \cup \bigcup_{i=n+1}^{2n-1} \left(P^{c'_i}_{c'_{i+1}} \right) \tag{2}$$

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

$$H = \bigcup_{i=1}^n \left(K^{c_i}_{v_{1,i}} \cup K^{v_{b_i,i}}_{c_{i+n}} \cup \bigcup_{j=2}^{b_i} K^{v_{j-1}}_{v_j} \right) \cup B_H \tag{4}$$

---

[1] If an instance of NMwTS does not allow the construction of the D^v_w gadgets, all values and the vector $\vec{b}$ are multiplied by 3.
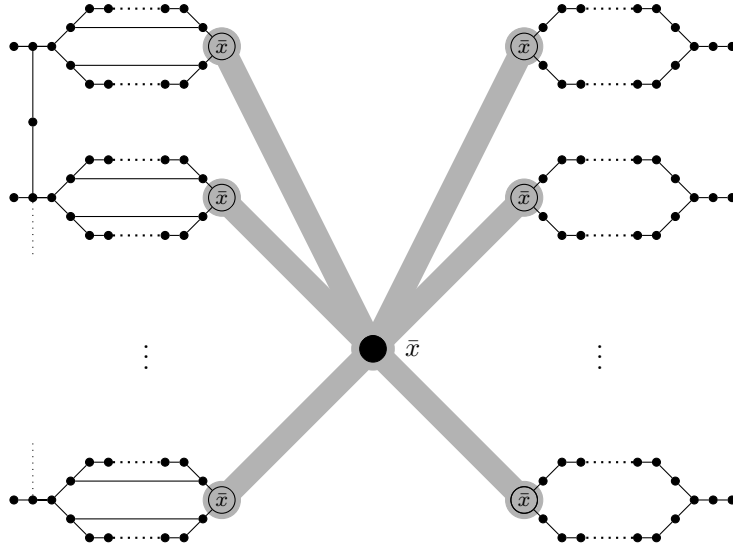
Figure 3: Base-gadget of G, where all vertices connected by the thick gray line represent the same vertex, i.e., vertex $\bar{x}$ with unbounded degree. There are n gadgets D_v^u on the left of $\bar{x}$ with the prime anchor vertices connected by $n-1$ anchor paths and n gadgets C_v^u on the right.

The graphs B_G and B_H (see Eq. 1 and 2 and for the former also Figure 3) are called *base-gadget* of G and H, resp. Each of these base-gadgets consist of n chordless cycles and n cycles with chords each, i.e., $C_{c_i}^{\bar{x}}$ and $D_{c_{i+n}}^{\bar{x}}$ for $i \in [n]$ in B_G and $C_{c_i}^{\bar{y}}$ and $D_{c_{i+n}}^{\bar{y}}$ for $i \in [n]$ in B_H. Hence, the base-gadgets of G and H contain one vertex with unbounded degree named $\bar{x}$ and $\bar{y}$, resp. The difference between the base-gadgets is that in B_G there are paths of length 2 (see P_w^v in Eq. 1 and 2) between the prime anchor vertices of the cycles containing chords and in B_H these paths are between the prime anchor vertices of the chordless cycles. Due to these paths, the resulting graphs are biconnected, as we will prove in Lemma 1. We call the paths between the prime anchor vertices *anchor paths*.

The base-gadgets are part of the graphs G and H and are later used to characterize the MCS of those graphs. The graph G represents the values of the elements in X and Y, see Figure 4. It contains n *xy-paths* one between each pair of anchor vertices c_i and c_{i+n} for all $i \in [n]$. The i-th xy-path delineates the values of the elements x_i and y_i. To this end, it consists of $s(x_i)$ connected K_w^v's (called *x-path*) which are connected to a path of length 2 (called *separating path*). In addition to the x-path, the separating path is also connected to $s(y_i)$ connected K_w^v's (called *y-path*). Analogously, H represents the values in the vector $\vec{b}$, see Figure 5. There is a *b-path* between each pair of anchor vertices c_i and c_{i+n} for all $i \in [n]$. Those b-paths represent the values of $\vec{b}$. For this purpose the i-th b-path consists of $b_i + 1$ gadgets K_w^v's.

We now show that both graphs G and H are series-parallel and have degree
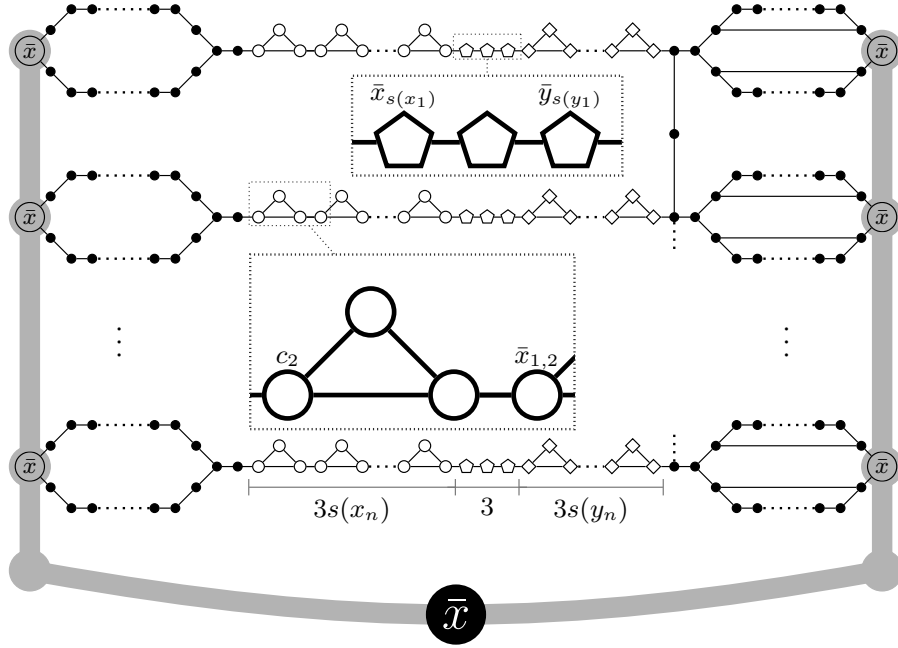
Figure 4: Graph G, containing the base-gadget as illustrated in Figure 3. All vertices connected by the thick gray line represent the same vertex $\bar{x}$. Circles represent x-paths, diamonds represent y-paths, where the i-th x- or y-path contains $3s(x_i)$ or $3s(y_i)$ vertices, resp. Pentagons represent the separation paths containing 3 vertices.

bounded by 3 for all but 1 vertex. We also show that both graphs can be computed in polynomial time with respect to the values of an NMwTS instance, which is necessary for the graphs to be used in a polynomial-time reduction.

Lemma 1. *The graphs G and H are biconnected.*

PROOF. Assume that G is 1-connected, but not 2-connected. Thus, there is a vertex $v \in V(G)$ such that v is a separator of G. Now there are three cases concerning the graph $G' := G \setminus \{v\}$:

1. If v is $\bar{x}$, then G' is still connected due to the anchor paths which connect all cycles containing chords which are again connected to the chordless cycles via the xy-paths.

2. If $v \neq \bar{x}$ is contained in a cycle of the base-gadget (in a C_w^v or D_w^v gadget), then G' is still connected as cycles are biconnected and all cycles contain $\bar{x}$ as common vertex.

3. If v is a vertex included in an anchor path or xy-path, then G' is still connected since each path is connected to two gadgets (of the same type, if v is present in an anchor path, and of different type otherwise) and even
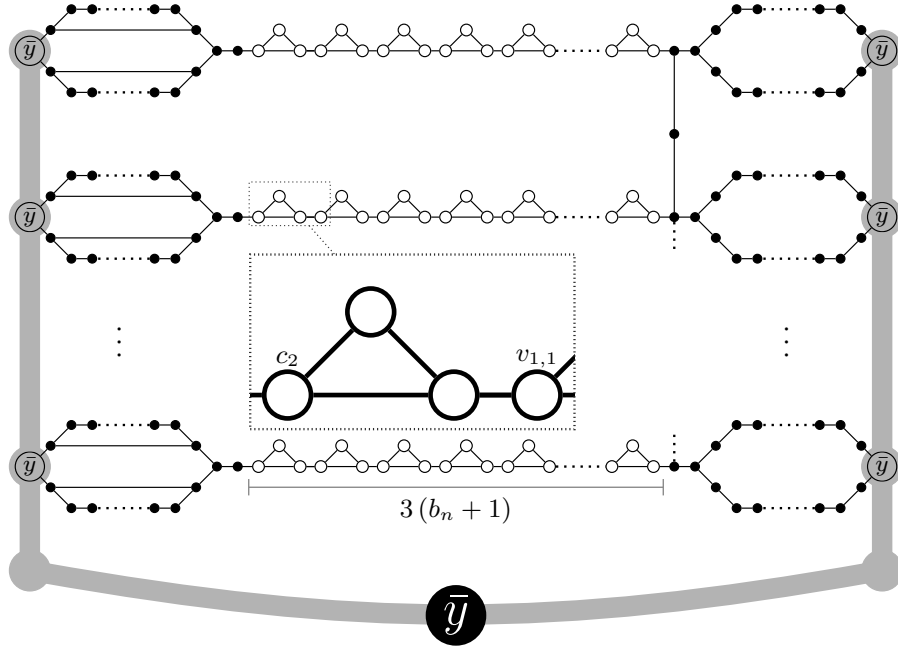
7

Figure 5: The graph H, containing a slightly different base-gadget as illustrated in Figure 3. All vertices connected by the thick gray line represent the same vertex $\bar{y}$. Circles represent b-paths containing $3(b_i + 1)$ vertices.

> if the path is split into two disjoint paths both of them would be connected to the base-gadget.

In none of these cases v is a separator and thus G is biconnected as the cycles in the base-gadget are biconnected. The same arguments can be applied to H if the xy-path is replaced with a b-path and the correct base-gadget is considered. □

We now sketch a proof which shows that the graphs are series-parallel. The idea is to show that G and H can be constructed starting with a finite set of $K_2^{s,t}$ only using the S- and P-operations described in Sec. 2. The important steps of the construction are visualized in Figure 6.

Lemma 2. *The graphs G and H are series-parallel.*

PROOF (SKETCH). 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 new s- and t-nodes. This process can be repeated until all xy- or b-paths are merged into their respective graphs. □

Since NMwTS is **NP**-complete in the strong sense [18], we are allowed to construct MCS instances for reduction with a size polynomial with respect to the

values of the NMwTS instance.

Lemma 3. *The graphs G and H can be constructed in polynomial time with respect to the values of the NMwTS instance.*

PROOF. We show that the number of vertices and edges in both graphs G and H is polynomial in the values of the NMwTS instance: Due to their construction, the size of both base-gadgets is equal. Hence, $|V(B_G)| = |V(B_H)| = 4n\Sigma_s + 7n$. G and H also contain xy- and b-paths. Therefore, $|V(G)| = 4n\Sigma_s + 7n + 3\Sigma_s + 3n$ and $|V(H)| = 4n\Sigma_s + 7n + 3\Sigma_{\bar{b}} + 3n$. Thus both sizes are polynomial in the values of an NMwTS instance. Since the degree is bounded by 3 for all but one vertex in both graphs, the number of edges in both graphs is polynomial in those values, too. Consequently, these graphs can be constructed in polynomial-time. □

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 which is also depicted in Figure 7.

Lemma 4. *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 $\min(|V(P)|, |V(P')|) - 1$.*

PROOF. Due to the 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 strictly fewer K_3 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. □

Since G and H only contain the base-gadget and n xy- or b-paths, we also have to consider the base-gadget in an MCS of G and H. We can show, that all vertices in the base-gadgets except the ones in the anchor paths are contained in the MCS.

Lemma 5. *Let B_G and B_K be the base-gadgets of G and H and $\bar{x}$ and $\bar{y}$ be the vertices whose degree is not bounded by 3 in B_G and B_H, resp. Then an MCS of B_G and B_H has size $|V(B_G)| - n + 1$ and $\bar{x}$ is mapped to $\bar{y}$.*

PROOF. Note that $|V(B_G)| = |V(B_H)|$. We first show that the vertices with unbounded degree ($\bar{x}$ in B_G and $\bar{y}$ in B_H) are mapped to each other and then show that we can predict the size of an MCS. Assume that $\bar{x}$ is not mapped to $\bar{y}$ but to $v \in V(B_H)$ such that $v \neq \bar{y}$ in a common subgraph F. Due to the construction, v has degree of at most 3 and $\bar{x}$ has degree $4n$. Therefore, not all of the cycles $\bar{x}$ is contained in (C_w^u and D_w^u type gadgets) can be contained in F. If $\bar{x}$ and $\bar{y}$ are mapped to each other, all cycles can be contained in a common subgraph, thus F cannot be an MCS.
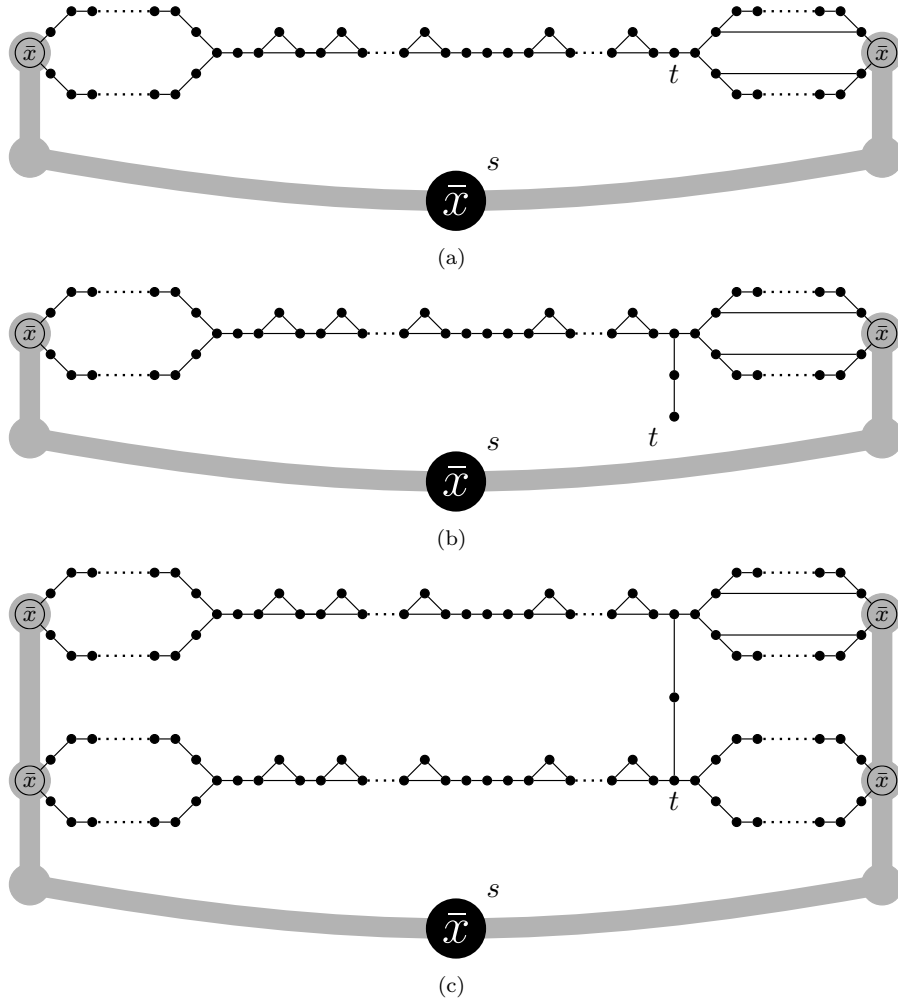
Figure 6: Construction of graph G only using the S- and P-operation. All vertices connected by the thick gray line represent the same vertex $\bar{x}$. (a) Two cycles and an xy-path connecting two anchor vertices, which is a series-parallel graph. (b) The same graph as before with an additional path attached to a prime anchor vertex which is still series-parallel. (c) A P-operation merging the s- and t-nodes of the graphs (a) and (b).

Now let F be a common subgraph of B_G and B_H where $\bar{x}$ is mapped to $\bar{y}$. In this case we only have to consider how the cycles of the base-gadgets are contained in F, i.e., how are the vertices in the cycles mapped in F. If a chordless cycle is mapped to a cycle containing chords, the following vertices cannot be contained in F: the anchor, prime anchor and the other vertex the prime anchor vertex is adjacent to in both cycles. If cycles are split up, i.e., one cycle is mapped to two other cycles, then there is at least one vertex per cycle which cannot be contained in F. Both cases result in at least n vertices
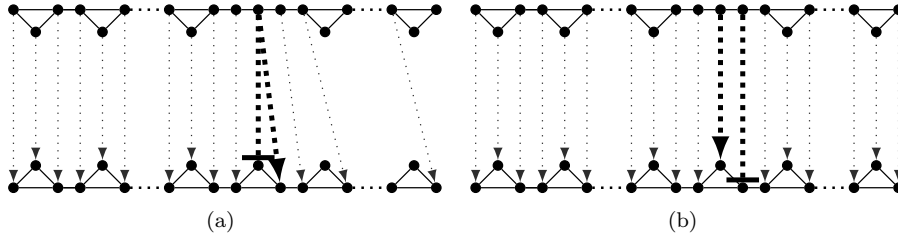
10

Figure 7: Possible mappings of xy-path (above) to a b-path (below) indicated by dashed arrows. In (a) the thick line with the bar as tip is not a feasible mapping, the only possibility is the thick arrow. As a result only 2 vertices of each following K_3 can be mapped. In (b) the x- and y-paths are completely mapped. Only two of the vertices in the separating path can be contained in the mapping as the thick line with the bar as tip is not a feasible mapping.

which are not contained in F. If however all cycles are mapped according to their gadget type, then only the vertices in the anchor paths cannot be in F. There are $n-1$ such vertices. Therefore, if all cycles are mapped that way, F is an MCS of size $|V(B_G)| - n + 1$. $\qquad\square$

3.2. Correctness of the Polynomial-Time Reduction

We have shown that G and H can be computed in polynomial time and that an MCS of G and H has some characteristics regarding its size and the vertices it contains. Concerning the polynomial-time 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 6. *An instance* $\left(X, Y, s, \vec{b}\right)$ *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. Let $\left(X, Y, s, \vec{b}\right)$ be an instance of NMwTS and G, H graphs constructed as described above. Assume that there is a numerical matching. Hence, $\Sigma_s = \Sigma_{\vec{b}}$ and thus $|V(G)| = |V(H)|$. An MCS of all xy-paths, b-paths and an MCS of the base-gadgets have size $|V(G)| - n$ and $|V(G)| - n + 1$ (Lemma 4, 5), resp. Even though they have been considered separately, the results can be combined, since all relevant vertices (the ones adjacent to the base-gadget and the ones in 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 (Lemma 4). The same is true for the base-gadgets, since the vertices of the anchor paths cannot be contained (Lem. 5). 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, we obtain the following result.

Theorem 7. $\mathrm{MCS}^{\leq 3,1}$ *in biconnected series-parallel graphs is* **NP**-*hard.*

4. Block-and-Bridge Preserving MCS in Series-Parallel Graphs

In this section we consider the task to find a maximum common subgraph that preserves the structure of blocks and bridges of the input graphs (BBP-MCS). This problem has been introduced in [11] and is also discussed in [9]. A common subgraph is said to be *block-and-bridge preserving* (BBP) if

BBP1 any two vertices in different blocks in the common subgraph must not be contained in the same block of an input graph, and

BBP2 each bridge in the common subgraph is a bridge in both input graphs.

In order to solve this problem, we make use of the following observation: Every two vertices in an input graph that are not in the same block cannot be in the same block in any common subgraph. Hence, together with conditions (BBP1) and (BBP2) we may infer that vertices in one block of G can only be mapped to vertices contained in exactly one block of H the resulting common subgraph is biconnected. This means that blocks and bridges can be considered separately to some extent and allows us to solve BBP-MCS by systematically decomposing the input graphs and solving the associated subproblems. We use the BC-tree data structure to decompose the connected input graphs into tree-like parts consisting of bridges and maximal biconnected subgraph, i.e., the blocks. Every block is then again decomposed by means of SP-trees. Solving MCS for the bridges that appear in the BC-tree is similar to the problem 1-MCS while the mapping between the blocks is closely related to 2-MCS. For this, we use the ideas of a recently proposed polynomial-time algorithm based on the SP-tree data structure [7]. Clearly, whenever two vertices of blocks are mapped that are both cutvertices of the input graphs, the components beyond these cutvertices must be considered for BBP-MCS.

4.1. Decomposing Series-Parallel Graphs

We consider simple connected series-parallel graphs and decompose them by means of BC- and SP-tree data structures. We refer to the vertices of SP- and BC-trees as *nodes* to distinguish them from the vertices of the input graphs. We use a notation and definition as introduced in [19] simplified and slightly modified for our application to series-parallel graphs. Let G be a simple connected graph, a *block* is a maximal biconnected subgraph of G and a *bridge* is an edge that is not contained in a block. Any two blocks and bridges of G may have at most one vertex in common, which must be a cutvertex.

Definition 3 (BC-tree). Given a connected graph G with at least two vertices, let C denote the set of cutvertices, Bl the set of blocks, Br the set of bridges and $B = Br \cup Bl$. The *BC-tree* $T^{\mathrm{BC}} = \mathrm{BC}(G)$ of G is the tree with nodes $B \cup C$ and edges between nodes $b \in B$ and $c \in C$ iff $c \in V(b)$.
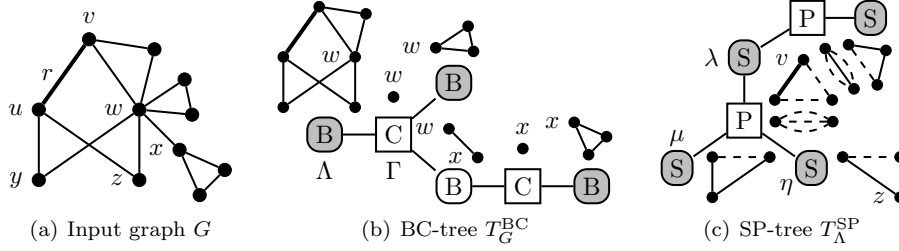
Figure 8: (a) A series-parallel graph and (b) its BC-tree with the associated skeleton graphs. Block B-nodes have a gray background color, while bridge B-nodes are not filled. (c) The SP-tree of the skeleton graph associated with the block B-node Λ together with its skeleton graphs.

We refer to the nodes of T^{BC} representing cutvertices as C-nodes and distinguish between bridge and block B-nodes; given a BC-tree T^{BC}, we refer to the three sets of nodes by $V_{\mathrm{C}}(T^{\mathrm{BC}})$, $V_{\mathrm{Br}}(T^{\mathrm{BC}})$ and $V_{\mathrm{Bl}}(T^{\mathrm{BC}})$, respectively. Each node Λ in a BC-tree has a skeleton graph S_Λ consisting of the vertices and edges of G represented by that node, i.e., a single vertex for C-nodes, a K_2 for bridge B-nodes and a biconnected subgraph for block B-nodes, cf. Figure 8.

SP-trees allow to decompose biconnected series-parallel graphs and can consequently be applied to the skeleton graphs of block B-nodes.

Definition 4 (SP-tree). Let G be a biconnected series-parallel graph with at least three vertices. The *SP-tree* $T^{\mathrm{SP}} = \mathrm{SP}(G)$ is the smallest tree that satisfies the following properties:

SP1 Each node 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 node types with the following skeleton structures:

    S: The skeleton S_λ is a simple chordless cycle, i.e., λ represents a series composition.

    P: The skeleton S_λ consists of two vertices and multiple parallel edges between them, i.e., λ represents a parallel composition.

SP3 For two adjacent nodes λ and η in T^{SP}, the skeleton graph S_λ contains a virtual edge e_η that represents 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 is exactly G.

The sets of S-nodes and P-nodes in T^{SP} are denoted by $V_S(T^{\mathrm{SP}})$ and $V_P(T^{\mathrm{SP}})$, respectively. The SP-tree T^{SP} is bipartite with respect to these two sets. Since a vertex v of a graph may occur in multiple skeleton graphs of the SP-tree T^{SP}, we denote by $\lambda(v)$ the representative of v in the skeleton S_λ. For the
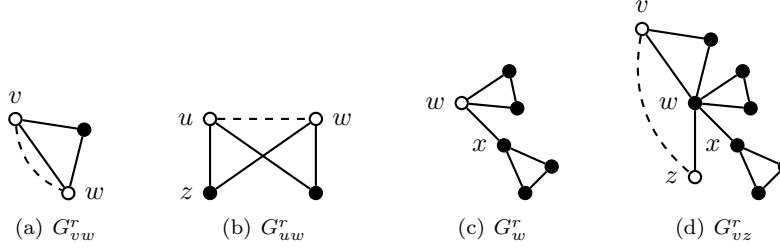
13

Figure 9: Some split graphs obtained from the graph depicted in Figure 8(a): In (a) and (b) the graph is split at different 2-separators, in (c) at a cutvertex and in (d) at a potential separator. Base vertices are not filled.

sake of simplicity we do not distinguish vertices of the original graph and their representatives in skeleton graphs, where this is clear from context. Let $r \in E(G)$, the SP-tree *rooted at r* is obtained by rooting T^{SP} at the node $\lambda \in V(T^{\mathrm{SP}})$ such that r is a real edge in S_λ. A rooted SP-tree induces a parent-child relation, where a node $\lambda \in V(T^{\mathrm{SP}})$ is the parent of an adjacent node $\eta \in V(T^{\mathrm{SP}})$ if the path from the root node to λ is shorter than the path from the root node to η. If a node $\lambda \in V(T^{\mathrm{SP}})$ is the parent of a node $\eta \in V(T^{\mathrm{SP}})$ and λ is the node pertinent to an edge $e_\lambda \in E(S_\eta)$, then e_λ is called a *reference edge* of λ and denoted by ref(λ). For the readers convenience, Greek upper- and lowercase letters denote the B-, C- and S-, P-nodes of the graph decompositions, respectively. Latin letters denote the vertices of the input graphs. Figure 8(c) shows an example of an SP-tree and the skeleton graphs associated with the individual nodes. It is well known that BC- as well as SP-trees can be computed in linear time [20].

Our BBP-MCS algorithm computes the solution based on subproblems for well-defined subgraphs of the input graphs closely related to these two data structures. Given a set of vertices S, let C_i^S denote the connected components of $G \setminus S$ and assume w.l.o.g. that at least one endpoint of a distinguished edge r is contained in C_1^S. Let $\mathcal{C}_S^r = \bigcup_{i \geq 2} V(C_i^S)$. We consider the following *split graphs*: Let $S = \{v\}$ with v a cutvertex, then G_v^r denotes the graph $G[\mathcal{C}_S^r \cup S]$ with *base vertex* v. Let $S = \{u, v\}$ be a minimal separator, then G_{uv}^r denotes the graph $G[\mathcal{C}_S^r \setminus (\mathcal{C}_{\{u\}}^r \cup \mathcal{C}_{\{v\}}^r) \cup S]$; we say u and v are the base vertices of G_{uv}^r. Figure 9 shows an example of several split graphs obtained from the graph depicted in Figure 8(a). When $e = (u, v)$ with $e \neq r$ is an edge that is part of a block, G_{uv}^r is defined as the edge e itself. Otherwise it is $(V, E \setminus e)$.

Note that there is a one-to-one correspondence between split graphs obtained from cutvertices and specific *rooted subtrees* of the BC-tree of G. Let G_v^r be a split graph, consider the C-node Γ representing the base vertex v and the unique B-node Λ with $r \in E(S_\Lambda)$, then the tree rooted at Γ, where the child subtree containing Λ is deleted, is associated with the split graph. For example, the split graph depicted in Figure 9(c) corresponds to the subtree of the BC-tree shown in Figure 8(b) that is obtained by rooting the tree at Γ and deleting Λ.

For handling block B-nodes we build on the algorithm described in [7], which

14

computes a maximum common biconnected subgraph of two series-parallel graphs. Again, the split graphs we consider are closely related to the data structure we use for the decomposition, i.e., SP-trees. The set $S = \{u, v\}$ is a (minimal) 2-separator of a biconnected graph if and only if both vertices are contained in the skeleton graph of an S-node and are not connected by a real edge. Since 2-separators are not sufficient to obtain correct solutions in biconnected series-parallel graphs, the concept of *potential separators* has been introduced. These are not only the 2-separators of the graph G, but also the 2-separators of an induced biconnected subgraph of G. In a series-parallel graphs all pairs of non-adjacent vertices that are contained in a chordless cycle are potential separators. For example, in Figure 8(a) the set $S = \{v, z\}$ is a potential separator, but not a separator of the graph, since u remains connected to w in $G \setminus S$ through the vertex y. Consequently, this vertex can not be contained in the subgraph that is separated by S. Removing these components systematically as detailed in [7] until a potential separator becomes a separator of the graph then allows to apply our notion of split graphs also in this case, cf. Figure 9(d). These split graphs can be obtained by means of SP-trees, where one vertex of a potential separator is contained in a skeleton graph of an S-node λ and the other in an S-node η. Removing all branches starting from P-nodes on the unique path with endpoints λ and η corresponds to the maximal biconnected subgraph that is separated by S. Consider the SP-tree shown in Figure 8(c), the vertex v is contained in S_λ and z in S_η and the component associated with the node μ consequently is removed to obtain a subgraph, which is then split to obtain G_{vz}^r.

4.2. The Algorithm

We now present a polynomial-time algorithm, which solves BBP-MCS in series-parallel graphs based on the decomposition of input graphs by means of BC- and SP-trees. We apply the idea presented in [15] for MCS in trees to the BC-trees constructing a solution based on smaller subproblems between rooted subtrees, which are combined with MwbM. In a similar manner the block B-nodes are handled by means of SP-trees, where certain additional technicalities must be considered [7].

The main procedure BBPMCS is presented as Algorithm 1. Given two series-parallel graphs as input, we assume that the associated BC- and SP-trees are already computed, which can be done in linear time. The algorithm computes the size of a BBP-MCS between the two input graphs by recursively dividing the problem into smaller subproblems defined between split graphs. The main procedure starts with pairs of B-nodes in the BC-trees of the input graphs. Since blocks and bridges must be preserved, only pairs of block B-nodes (Line 2–10) and pairs of bridge B-nodes (Line 11–16) are considered. In the former case, the algorithm loops over all possible pairs of S-nodes in the SP-trees of the two blocks. Two distinguished edges serve as starting point of the mapping by fixing the mapping of their endpoints. The resulting subproblem then is handled by the procedure SERIES, see Algorithm 2. The main procedure does not take into account that the vertices u and u' (or v and v') may be cutvertices

---

Algorithm 1: BbpMcs(G,H)

    Input   : Two series-parallel graphs G and H.
    Output : Size of a BBP-MCS of the graphs G and H.
    Data   : BC-trees T_G^{BC} and T_H^{BC} of G and H, respectively, SP-trees
              T_Λ^{SP} for every block B-node Λ in T_G^{BC} and T_H^{BC}.

1 $mcs \leftarrow 0$
2   **forall** $(\Lambda, \Lambda') \in V_{\mathrm{Bl}}(T_G^{\mathrm{BC}}) \times V_{\mathrm{Bl}}(T_H^{\mathrm{BC}})$ **do** $\triangleright$ *Pairs of block B-nodes*
3   ROOT($T_G^{\mathrm{BC}}, \Lambda$); ROOT($T_H^{\mathrm{BC}}, \Lambda'$)
4   **forall** $(\lambda, \lambda') \in V_S(T_\Lambda^{\mathrm{SP}}) \times V_S(T_{\Lambda'}^{\mathrm{SP}})$ **do** $\triangleright$ *Pairs of S-nodes*
5 $r \leftarrow$ arbitrary $(u,v) \in E(S_\lambda) \cap E(G)$; ROOT($T_\Lambda^{\mathrm{SP}}, r$)
6   **forall** *edges* $r' = (u', v') \in E(S_{\lambda'}) \cap E(H)$ **do**
7   ROOT($T_{\Lambda'}^{\mathrm{SP}}, r'$)
8 $p \leftarrow$ SERIES(u, v, u', v')
9 $p \leftarrow$ SERIES(u, v, v', u') $\triangleright$ *Alternative edge mapping*
10 $mcs \leftarrow \max\{mcs, p, q\}$

11   **forall** $(\Lambda, \Lambda') \in V_{\mathrm{Br}}(T_G^{\mathrm{BC}}) \times V_{\mathrm{Br}}(T_H^{\mathrm{BC}})$ **do** $\triangleright$ *Pairs of bridge B-nodes*
12   ROOT($T_G^{\mathrm{BC}}, \Lambda$); ROOT($T_H^{\mathrm{BC}}, \Lambda'$)
13 $r = (u,v) \leftarrow E(S_\Lambda^{\mathrm{BC}})$; $r' = (u', v') \leftarrow E(S_{\Lambda'}^{\mathrm{BC}})$
14 $p \leftarrow$ CUT(u, u') + CUT(v, v')
15 $q \leftarrow$ CUT(u, v') + CUT(v, u') $\triangleright$ *Alternative edge mapping*
16 $mcs \leftarrow \max\{mcs, p, q\}$
17 **return** $mcs + 2$

---

or that the edges r and r' could be virtual. These cases are not considered since the possible better result would be obtained anyway for a different starting configuration. In the latter case, the edges associated with the bridge B-nodes serve as starting point of the mapping. Again, the two possible mappings of endpoints are considered and the solution is determined based on the return values of CUT, see Algorithm 4. Assume we have the bridges $r = (u, v)$ of G and $r' = (u', v')$ of H and map the vertex u to u' and v to v'. The result is obtained by computing the BBP-MCS between the split graphs G_u^r and $G_{u'}^{r'}$ and the result for the opposing split graphs G_v^r and $G_{v'}^{r'}$. Finally, the procedure returns the best solution found and accounts for the two base vertices, which have been mapped, by adding two. Whenever we compute the BBP-MCS between two split graphs, we require that the mapping of base vertices is fixed and compute the maximum possible solution under this constrain. We will give the details of the procedures called by BbpMcs in the following.

The procedure SERIES essentially computes the 2-MCS between two blocks based on their SP-trees as in [7, Mcs-S], but is modified to take cutvertices into account. Whenever the procedure is called the SP-trees considered have been rooted at some edge and we may make use of the parent-child relationship between adjacent nodes. For a virtual edge e in the skeleton S_λ of an S-node

---

Algorithm 2: $\text{SERIES}(u, v, u', v')$

> **Input** : Base vertices u, v of G, $v \in V(S_\lambda)$ and u', v' of H,
> $v' \in V(S_{\lambda'})$.
>
> **Output :** Size of a BBP-MCS between G^r_{uv} and $H^{r'}_{u'v'}$ under the
> restriction that u is mapped to u' and v to v'.

1 $e = (v, w) \leftarrow \text{NEXT}(v, S_\lambda)$ ▷ *Next edge in S_λ*
2 $e' = (v', w') \leftarrow \text{NEXT}(v', S_{\lambda'})$ ▷ *Next edge in $S_{\lambda'}$*
3 if $e = \text{ref}(\lambda)$ **then return** $\text{SERIES}(u, \text{pS}(v), u', v')$ ▷ *Parent S-node*
4 if $e' = \text{ref}(\lambda')$ **then return** $\text{SERIES}(u, v, u', \text{pS}(v'))$ ▷ *Parent S-node*
5 if $w = u$ **and** $w' = u'$ **then**   ▷ *Completed skeleton*
6   **return** $\text{EDGE}(v, w, v', w')$

7 if $w = u$ **xor** $w' = u'$ **then return** $-\infty$ ▷ *Incompletable mapping*
8 $mcs \leftarrow \text{EDGE}(v, w, v', w') + \text{CUT}(w, w') + \text{SERIES}(u, w, u', w') + 1$
9 if $e \notin E(G)$ **or** $e' \notin E(H)$ **then**   ▷ *Consider potential separators*
10   **if** $e \in E(G)$ **then** $M \leftarrow \{\lambda\}$
11   **else** $M \leftarrow \text{cS}(e)$
12   **if** $e' \in E(H)$ **then** $M' \leftarrow \{\lambda'\}$
13   **else** $M' \leftarrow \text{cS}(e')$
14   **forall** $(\eta, \eta') \in M \times M'$ **do**
15 $p \leftarrow \text{SERIES}(u, \eta(v), u', \eta'(v'))$ ▷ *Continue in child S-node*
16 $mcs \leftarrow \max\{mcs, p\}$

17 return mcs

λ we denote the children of the P-node pertinent to e by $\text{cS}(e)$. For a vertex v that is one endpoint of the reference edge in a skeleton S_μ we refer to the representative of v in the next S-node on the path to the root by $\text{pS}(v)$. The function $\text{NEXT}(u, S_\lambda)$ returns the vertex adjacent to u which yet has not been considered in the skeleton graph of λ. To obtain a BBP-MCS the common subgraph of two blocks must be biconnected. Hence, we simultaneously traverse the cyclic skeleton graphs of two S-nodes, one of T_G^{SP} and the other of T_H^{SP}, starting from one endpoint of an edge in each. Eventually, we have to return to the other endpoints of these edges in both graphs at the same time. In each step the mapping is extended by the next vertex w lying on the cycle starting from the second base vertex v. In the case that the edge (v, w) is virtual, the edge represents parts of the graph beyond this edge, which are considered by the procedure EDGE. If the next vertex w is a cutvertex in the input graph, the parts of the graphs that are not contained in the block are taken into account by the procedure CUT. Lines 9 to 16 handle potential separators, where the two base vertices are contained in different S-nodes.

     We illustrate the procedure and how the approach aligns with our definition of split graphs in the following. Consider the example shown in Figure 8(a) and assume that the main procedure BBPMCS has started with a pair of block B-nodes, where the first is $\Lambda \in V(T_G^{\text{BC}})$. Further, assume that the S-node λ has

---

Algorithm 3: EDGE(u, v, u', v')

Input : Vertices with $e = (u, v) \in E(S_\lambda)$ and $e' = (u', v') \in E(S_{\lambda'})$.
Output : Size of a BBP-MCS of G^r_{uv} and $H^{r'}_{u'v'}$ under the restriction
   that u is mapped to u' and v to v'.

1 **if** $e \in E(G)$ **xor** $e' \in E(H)$ **then** **return** $-\infty$ ▷ *Subgraph not induced*
2 **if** e *is real in* S_μ **or** e' *is real in* $S_{\mu'}$ **then** **return** 0 ▷ *Valid mapping*
3 $M \leftarrow \mathrm{cS}(e)$; $M' \leftarrow \mathrm{cS}(e')$
4 **forall** $m = (\eta, \eta') \in M \times M'$ **do**   ▷ *Pairs of S-node children*
5 $\quad\lfloor\ w(m) \leftarrow \mathrm{SERIES}(\eta(u), \eta(v), \eta'(u'), \eta'(v'))$

6 $p \leftarrow \mathrm{MWBMATCHING}(M, M', w)$ ▷ *Compute maximum weight matching*
7 **if** $p \neq 0$ **or** $e \in E(G)$, $e' \in E(H)$ **then** **return** p
8 **else** **return** $-\infty$ ▷ *Not biconnected*

---

been selected from the SP-tree associated with the block, which has been rooted at the edge r. Then, the next vertex in S_λ is w, which is connected to the base vertex v by the virtual edge (v, w). The situation corresponds to the case that is handled in Line 8 of the procedure SERIES. The problem here is divided into three subproblems: First, if possible, the part of the input graph represented by the virtual edge must be mapped, i.e., the split graph G^r_{vw}, see Figure 9(a). This is done by the procedure EDGE, explained in detail later. Second, the vertex w is a cutvertex in G and the split graph G^r_w, see Figure 9(c), must be taken into account whenever w is mapped to a cutvertex in H. This is handled by the procedure CUT. Finally, the remainder of the skeleton S_λ must be considered, which is done by the recursive call of SERIES, where the vertex w replaces v as base vertex. Consequently, this procedure tries to map the split graph G^r_{uw}, see Figure 9(b). Since the vertex w is mapped, one is added to the sum over the return values of these procedures.

The two procedures EDGE and CUT both construct MwbM instances to determine the matching between unmapped connected components obtained from a 2-separator and a cutvertex, respectively. Algorithm 3 gives the details of EDGE, which is called with the endpoints of two edges as arguments, one in the skeleton of the S-node λ and the other in the skeleton of λ'. Note that it is possible that there is a virtual edge (u, v) in a skeleton graph, although (u, v) is not an edge of the input graph. If only one of the edges is contained in the input graph, the mapping is not allowed since we would not obtain a common induced subgraph. If both edges are virtual, then the endpoints are separators, see (SP1), and the BBP-MCS of the split graph has to be added to the result. This is done by computing the maximum weight matching in the complete bipartite graph C, where the two vertex sets are the children of the P-node pertinent to e and e', respectively. The weight $w \colon E(C) \to \mathbb{N} \cup \{-\infty\}$ of an edge $e = (\lambda, \lambda')$, where $\lambda \in V_S(T^{\mathrm{SP}}_G)$ and $\lambda' \in V_S(T^{\mathrm{SP}}_H)$, is the size of a BBP-MCS between the two components associated with its endpoints. This value is determined by the procedure SERIES starting from the reference edges

---

Algorithm 4: $\text{CUT}(u, u')$

Input : Two vertices $u \in V(G)$, $u' \in V(H)$.
Output : Size of a BBP-MCS between G_u^r and $H_{u'}^{r'}$ under the
restriction that u is mapped to u'; 0 if u and u' not both are
cutvertices.

1 **if** u *and* u' **not** *both are cutvertices* **then return** 0
2 $M \leftarrow \text{cB}(u)$; $M' \leftarrow \text{cB}(u')$ ▷ *Get B-node children*
3 **forall** $m = (\Lambda, \Lambda') \in M \times M'$ **do**
4   **if** Λ *and* Λ' *both are bridge B-nodes* **then**
5 $(u, v) \leftarrow E(S_\Lambda)$; $(u', v') \leftarrow E(S_{\Lambda'})$ ▷ *Get associated bridges*
6 $w(m) \leftarrow \text{CUT}(v, v') + 1$
7   **else if** Λ *and* Λ' *both are block B-nodes* **then**
       ▷ *Get the S-nodes containing an edge incident to base vertices*
8 $N \leftarrow \{\lambda \in V_S(T_\Lambda^{\text{SP}}) \mid \exists r = (u, v) \in E(S_\lambda) \cap E(G)\}$
9 $N' \leftarrow \{\lambda' \in V_S(T_{\Lambda'}^{\text{SP}}) \mid \exists r' = (u', v') \in E(S_{\lambda'}) \cap E(H)\}$
10   **forall** $(\lambda, \lambda') \in N \times N'$ **do**   ▷ *Pairs of relevant S-nodes*
11 $r \leftarrow$ arbitrary $(u, v) \in E(S_\lambda) \cap E(G)$; $\text{ROOT}(T_\lambda^{\text{SP}}, r)$
12   **forall** *edges* $r' = (u', v') \in E(S_{\lambda'}) \cap E(H)$ **do**
13 $\text{ROOT}(T_{\lambda'}^{\text{SP}}, r')$
14 $w(m) \leftarrow \max\{w(m), \text{SERIES}(u, v, u', v')\}$

15   **else** $w(m) \leftarrow -\infty$ ▷ *Non BBP matching*
16 **return** $\text{MWBMATCHING}(M, M', w)$

---

of the S-nodes λ and λ'. Note that if the result of the matching is 0 and there is no edge between the base vertices, there is no path between the base vertices through the split graph considered. Since in this case the common subgraph would not be biconnected, contradicting (BBP2), the value $-\infty$ is returned (Line 8).

The procedure CUT computes the size of a BBP-MCS of two split graphs obtained from cutvertices. Therefore, zero is returned if the given vertices u and u' are not both cutvertices. Otherwise, we consider their child B-nodes $\text{cB}(u)$ and $\text{cB}(u')$ in the rooted BC-trees. To this end, we again create a weighted complete bipartite graph C with vertex partition $\text{cB}(u) \cup \text{cB}(u')$. The weight $w \colon E(C) \to \mathbb{N} \cup \{-\infty\}$ of an edge is the size of a BBP-MCS of the two split graphs associated with its endpoints. Note that the node sets contain block B-nodes as well as bridge B-node. Edges connecting different types of B-nodes obtain weight $-\infty$ (Line 15) as mapping them contradicts restriction (BBP1). This assures that these edges are not contained in any maximum weight matching. The weight of an edge between two bridge B-nodes is determined by a recursive call of CUT, where the current base vertices are replaced by the other endpoints of the bridges. In case of two block B-nodes the BBP-MCS is determined in a similar manner as in the main procedure, with the difference that

only S-nodes are considered that contain a representative of the base vertex with an appropriate incident edge $r \in E(G)$, which is mapped to an edge $r' \in E(H)$. These two edges serve as roots of the SP-trees and for each pair the procedure SERIES is called. The maximum value returned by any of these calls yields the edge weight. Note that the mapping of the base vertices is fixed and — in contrast to the main procedure — we do not need to consider all the possible mappings.

4.3. Analysis

We analyze Algorithm 1 and show that BBP-MCS can be solved in polynomial time. We give improved bounds on the running time for the case that both input graphs are outerplanar.

Theorem 8. *The problem BBP-MCS in series-parallel graphs can be solved in time $\mathcal{O}(n^6)$, where n is the number of vertices in the larger input graph.*

PROOF. The correctness of the algorithm is based on the argumentation above and the argumentation in [7]. To prove the running time, we transform the algorithm in a dynamic programming approach. In [7, Theorem 1] it is shown, that 2-MCS can be solved in time $\mathcal{O}(n^6)$ when storing the 2-MCS between all pairs of split graphs in a table of size $\mathcal{O}(n^4)$. Note that we also have to consider split graphs that are obtained from cutvertices, for which another table of size $\mathcal{O}(n^2)$ is sufficient.

We consider the running time required by the procedure SERIES. Assume that for each split graph that is smaller than the split graph considered by the current call, the BBP-MCS has been computed. Then all calls to procedures are answered in constant time by a lookup in the table and the running time for calling SERIES once is $\mathcal{O}(n^2)$. Since in each call a cell of the table is filled the total running time caused by all calls of this procedure is $\mathcal{O}(n^6)$.

The procedures CUT and EDGE both require to solve MwbM problems. In case of CUT, each matching problem corresponds to a pair of rooted subtrees of the BC-trees. There can be at most $\mathcal{O}(n^2)$ such pairs and each matching problem can be solved in $\mathcal{O}(n^3)$ by the Hungarian method. This results in a total running time of $\mathcal{O}(n^5)$. For the procedure EDGE, each matching problem corresponds to a pair of subtrees of the SP-trees. Each subtree is obtained for some P-node, where one adjacent S-node is removed. With the same argument as above the total running time of the procedure is $\mathcal{O}(n^5)$ as in [7]. Finally, we consider the main procedure BBPMCS. Since we may assume that all calls to procedures are answered in constant time by lookups in the tables, the running time of the procedure is $\mathcal{O}(n^2)$. Consequently, the total running time of the algorithm is dominated by the procedure SERIES and is $\mathcal{O}(n^6)$. □

If we restrict to outerplanar graphs, each P-node in the SP-trees has degree two which concludes in the following theorem.

Theorem 9. *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 8. Since all P-nodes in SP-trees of outerplanar graphs have degree two, the total running time of SERIES reduces to $\mathcal{O}(n^4)$, since the sets M and M' considered in Line 14 contain only one element. Moreover, there is no need to use MwbM in the procedure EDGE as the bipartite graphs are K_2's. Consequently the running time of a single call of EDGE becomes constant. For the procedure CUT the restriction to outerplanar graphs does not allow improved bounds on the running time since the number of rooted subtrees of the BC-tree does not change. Therefore, the total running time is $\mathcal{O}(n^5)$. □

It is known that BBP-MCES in outerplanar graphs can be solved in polynomial time [11, 1, 12, 6], where MCES refers to a variation of the problem that asks for edge-induced common subgraphs with maximum number of edges. 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 an MCES, but not an MCS. In [6] the rough bound of $\mathcal{O}(n^{10})$ for an BBP-MCES algorithm was given. The approach by Schietgat et al. has been shown to be efficient in practice [12]. Different worst-case bounds are provided in several publications starting with $\mathcal{O}(n^7)$ [11], then $\mathcal{O}(n^5)$ [1] and finally $\mathcal{O}(n^2\sqrt{n})$ [12].

5. Concluding Remarks

We showed that MCS in series-parallel graphs with degree bounded by 3 for all but one vertex is **NP**-hard by reduction from NMwTS. Then extended a 2-MCS algorithm [7] to solve BBP-MCS with running time $\mathcal{O}(n^6)$. In outerplanar graphs, it can solve BBP-MCS in $\mathcal{O}(n^5)$. BBP-MCES in outerplanar graphs was taken as basis to obtain polynomial time solutions for MCES in outerplanar graphs of bounded degree [9] and it has yet to be decided whether MCES in series-parallel graphs of bounded degree can be solved in polynomial time. To the authors' 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* [21]. It still is unknown whether MCS or MCES in series-parallel graphs is solvable in polynomial time if all vertices have bounded degree. Since 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**-hard for $k \geq 11$ even when the degree is bounded [6]. For all other $k > 1$, the complexity has yet to be decided.

References

References

[1] L. Schietgat, J. Ramon, M. Bruynooghe, H. Blockeel, An efficiently computable graph-based metric for the classification of small molecules, in: Discovery Science, Vol. 5255 of LNCS, Springer Berlin / Heidelberg, 2008, pp. 197–209.

[2] T. Horváth, J. Ramon, Efficient frequent connected subgraph mining in graphs of bounded tree-width, Theoretical Computer Science 411 (31–33) (2010) 2784 – 2797. doi:10.1016/j.tcs.2010.03.030.

[3] J. Matoušek, R. Thomas, On the complexity of finding iso- and other morphisms for partial k-trees, Discrete Mathematics 108 (1-3) (1992) 343–364.

[4] A. Gupta, N. Nishimura, Sequential and parallel algorithms for embedding problems on classes of partial k-trees, in: SWAT '94, Vol. 824 of LNCS, 1994.

[5] A. Gupta, N. Nishimura, The complexity of subgraph isomorphism for classes of partial k-trees, Theoretical Computer Science 164 (1-2) (1996) 287–298.

[6] T. Akutsu, T. Tamura, On the complexity of the maximum common subgraph problem for partial k-trees of bounded degree, in: Algorithms and Computation, Springer, 2012, pp. 146–155.

[7] N. Kriege, P. Mutzel, Finding maximum common biconnected subgraphs in series-parallel graphs, in: MFCS, Vol. 8635 of LNCS, Springer, 2014, pp. 505–516.

[8] T. Akutsu, A polynomial time algorithm for finding a largest common subgraph of almost trees of bounded degree, IEICE Trans. Fundamentals E76-A (9).

[9] T. Akutsu, T. Tamura, A polynomial-time algorithm for computing the maximum common connected edge subgraph of outerplanar graphs of bounded degree, Algorithms 6 (1) (2013) 119–135.

[10] M. M. Sysło, The subgraph isomorphism problem for outerplanar graphs, Theoretical Computer Science 17 (1) (1982) 91 – 97. doi:10.1016/0304-3975(82)90133-5.

[11] L. Schietgat, J. Ramon, M. Bruynooghe, A polynomial-time metric for outerplanar graphs, in: Mining and Learning with Graphs (MLG), 2007.

[12] L. Schietgat, J. Ramon, M. Bruynooghe, A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics, Annals of Mathematics and Artificial Intelligence 69 (4) (2013) 343–376. doi:10.1007/s10472-013-9335-0.

[13] L. Schietgat, F. Costa, J. Ramon, L. De Raedt, Effective feature construction by maximum common subgraph sampling, Machine Learning 83 (2) (2011) 137–161. doi:10.1007/s10994-010-5193-8.

[14] A. Brandstädt, V. B. Le, J. P. Spinrad, Graph Classes: A Survey, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[15] D. W. Matula, Subtree isomorphism in $O(n^{5/2})$, in: Algorithmic Aspects of Combinatorics, Vol. 2 of Ann. Discrete Math., 1978, pp. 91 – 106.

[16] H. W. Kuhn, The Hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2) (1955) 83–97.

[17] M. R. Garey, D. S. Johnson, "Strong" NP-completeness results: Motivation, examples, and implications, Journal of the ACM (JACM) 25 (3) (1978) 499–508.

[18] M. R. Garey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, WH Freeman and Company, New York, 1979.

[19] M. Chimani, P. Hliněnỳ, A tighter insertion-based approximation of the crossing number, in: Automata, Languages and Programming, Springer, 2011, pp. 122–134.

[20] C. Gutwenger, P. Mutzel, A linear time implementation of SPQR-trees, in: J. Marks (Ed.), Graph Drawing, Vol. 1984 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 77–90.

[21] T. Nishizeki, J. Vygen, X. Zhou, The edge-disjoint paths problem is np-complete for series–parallel graphs, Discrete Applied Mathematics 115 (1) (2001) 177–186.