Finding minimum Tucker submatrices

Ján Maňuch *and Arash Rafiey †

Abstract

A binary matrix has the Consecutive Ones Property (C1P) if its columns can be ordered in such a way that all 1s on each row are consecutive. These matrices are used for DNA physical mapping and ancestral genome reconstruction in computational biology, on the other hand, they represent a class of convex bipartite graphs and are of interest of algorithm graph theory researchers. Tucker gave a forbidden submartices characterization of matrices that have the C1P property in 1972. Booth and Lucker (1976) gave a first linear time recognition algorithm for matrices with C1P property, and then in 2002, Habib, et al. gave a simpler linear time recognition algorithm. There has been substantial amount of works on efficiently finding minimum size forbidden submatrix. Our algorithm is at least n times faster than the existing algorithm where n is the number of columns of the input matrix.

1 Introduction and Preliminaries

A binary matrix has the Consecutive Ones Property (C1P) if its columns can be ordered in such a way that all ones in each row are consecutive. Deciding if a matrix has the C1P can be done in linear-time and linear space [4, 7, 8, 12, 13]. The problem of deciding if a matrix has the C1P has been considered in genomic, for problems such as physical mapping [2, 9] or ancestral genome reconstruction [1, 5, 11].

Let M be a $m \times n$ binary matrix. Let $\mathbf{R} = \{r_i : \ i = 1, \ldots, m\}$ be the set of its rows and $\mathbf{C} = \{c_j : \ j = 1, \ldots, n\}$ the set of its columns. Its *corresponding bipartite graph* $G(M) = (V_M, E_M)$ is defined as follows: $V_M = \mathbf{R} \cup \mathbf{C}$, and two vertices $r_i \in R$ and $c_j \in \mathbf{C}$ are connected by an edge if and only if $M[i, j] = 1$. We will refer to the partition $\mathbf{R}$ and $\mathbf{C}$ of G as black and white vertices, respectively. A bipartite graph $G = (B, W)$ with black (B) and white (W) vertices is called *convex with respect to* W; if there exists a linear ordering of the white vertices such that the neighborhood of each black vertex is an interval, i.e. consecutive. A binary matrix M has the C1P property if and only if $G(M) = (R, W)$ is a convex bipartite graph with respect to W.

In the rest of this paper we consider convexity with respect to the white vertices unless we specify.

For a graph $H = (V, E)$, the set of neighbors of a vertex $x \in V(H)$ will be denoted by $N(x)$. The i-the neighborhood of x, denoted by $N_i(x)$, is the set of vertices at distance i

*University of British Coloumbia and Simon Fraser University, BC,Canada, Eamil: jmanuch@cs.ubc.ca
†Simon Fraser University, BC, Canada and Indiana State University, IN, USA, Email: arashr@sfu.ca

from x. All these sets, for a fixed x, can be computed in time $O(e)$ using the bread-first search algorithm where e denotes the number of edges of H. A subgraph of H induces by vertices $x_1, \ldots, x_k$ will be denoted by $H[x_1, \ldots, x_k]$.

For simplicity, we denote the edge u, v of a graph H by uv .

Let G be a bipartite graph. A subset M of the edges G is called an *induced matching* if $G[V(M)]$ is isomorphic to M. Here $V(M)$ is the endpoints vertices of the edges in M and $G[V(M)]$ is the induced subgraph of G by $V(M)$. For example, two edges $uv, u'v'$ of G where u, u' are in the same partition form an induced matching if $uv', u'v$ are not edges of G.

An *asteroidal triple* is an independent set of three vertices such that each pair is connected by a path that avoids (does not go through) the neighborhood of the third vertex. A *white asteroidal triple* is an asteroidal triple on white (column) vertices.

The following result of Tucker links the C1P of matrices to asteroidal triples of their bipartite graphs.

Theorem 1 ([14]). *A binary matrix has the C1P if and only if its corresponding bipartite graph does not contain any white asteroidal triples.*

Theorem 2 ([14]). *A binary matrix has the C1P if and only if its corresponding bipartite graph does not contain any of the forbidden subgraphs in $T = \{G_{\mathrm{I}_k}, G_{\mathrm{II}_k}, G_{\mathrm{III}_k} : k \geq 1\} \cup \{G_{\mathrm{IV}}, G_{\mathrm{V}}\}$, depicted in Figure 1. We will refer to these subgraphs as type I, II, III, IV and V, respectively.*

The author in [10] developed an algorithm for finding one of the obstructions in linear time. However, their algorithm does not guarantee to find the minimum size obstruction. The characterization can be used to determine whether a given binary matrix M has the C1P in time $O(\Delta mn^2 + n^3)$, where Δ is the maximum number of ones per row, i.e., the maximum degree of the black vertices in $G(M)$, as explained by the following result in [6].

Lemma 1 ([6]). *A white asteroidal triple u, v, w with the smallest sum of the three paths can be computed in time $O(\Delta mn^2 + n^3)$.*

For practical purposes, there is a much faster algorithm that uses PQ-trees for determining whether a binary matrix has the C1P, cf. [4].

One of the reasons that the authors in [6] were in these obstructions was to developed faster approximation algorithms and fixed-parameterized algorithms for consecutive ones submatrix problems. *Our goal in this paper is to improve the algorithms in [3, 6].*

2 Summary of known results and new results

We first summarize the running time of the previous known algorithms for finding each of the forbidden subgraphs. Let G be a bipartite graph. We denote the maximum degree of G by Δ.

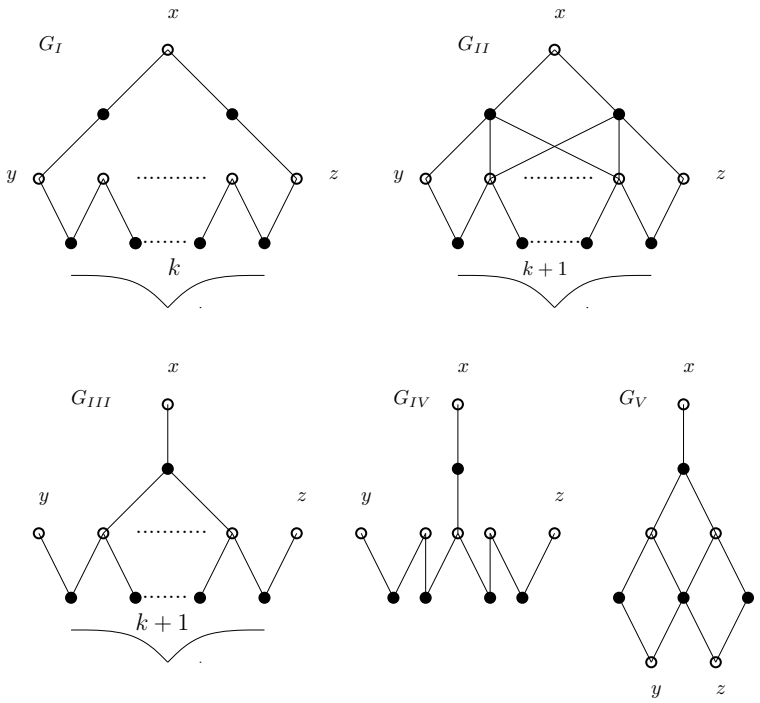

Figure 1: The set of Tucker's forbidden subgraphs.

Known Results: The known results regarding how fast one can find one of the obstructions to a matrix with the $C1P$ property can be summarize in the following theorem.

Theorem 3. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then we have the following.*

1. *One can find a minimum size induced subgraph G_I, (type I obstruction) in G in time $\mathcal{O}(\Delta^4 m^3)$ if one exists [3].*
2. *One can find a minimum size induced subgraph G_{II}, (type II obstruction) of G in time $\mathcal{O}(\Delta^4 m^3) = \mathcal{O}(ne^3)$ [3].*
3. *One can find a minimum size induced subgraph G_{III}, (type III obstruction) of G in time $\mathcal{O}(\Delta^2 m^2 n^2)$ if one exists [3].*
4. *One can find a minimum size induced subgraph G_{IV}, (type IV obstruction) of G in time $\mathcal{O}(\Delta^3 m^2 n^3)$ if one exists [6].*
5. *One can find a minimum size induced subgraph G_V, (type V obstruction) of G in time $\mathcal{O}(\Delta^4 m^2 n^2)$ if one exists [6].*

Now combining the results from [3] and [6], we have the following theorem.

Theorem 4. *Let $G = (B, W)$ be a bipartite graph with m black vertices and n white vertices. Then one can find a minimum size obstruction to G being convex (respect to the white vertices), in time $\mathcal{O}(\Delta^3 m^2(\Delta m + n^3))$.*

Our Results : We will consider two problems: (1) detecting a smallest forbidden subgraph of each type (Section 3.1), and (2) detecting a smallest forbidden subgraph of any type (Section 3.2).

Theorem 5. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then we have the following.*

1. *One can find a minimum size induced subgraph G_I, (type I obstruction) in G in time $\min\{\mathcal{O}(\Delta e^2), \mathcal{O}(\Delta^3 m^2)\}$ if one exists.*
2. *One can find a minimum size induced subgraph G_{III}, (type III obstruction) of G in time $\min\{\mathcal{O}(e^3), \mathcal{O}(\Delta^3 m^3)\}$ if one exists.*
3. *One can find a minimum size induced subgraph G_{IV}, (type IV obstruction) of G in time $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$ if one exists.*
4. *One can find a minimum size induced subgraph G_V, (type V obstruction) of G in time $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$ if one exists.*

We further can refine Theorem 5 and obtain the following theorem.

Subgraph type	Time complexity		
	Previous result	Our result (Exact)	Our result
I	$O(\Delta^4 m^3)$ [3]	$O(\Delta e^2) = O(\Delta^3 m^2)$	$O(n^2 e)$ [6]
II	$O(\Delta^4 m^3) = O(ne^3)$ [3]	—	$O(n^2 e)$ [6]
III	$O(\Delta^2 m^2 n^2)$ [3]	$O(e^3) = O(\Delta^3 m^3)$	$O(ne^2)$
IV	$O(\Delta^3 m^2 n^3)$ [6]	$O(m^3 e) = O(\Delta m^4)$	$O(n^3 e)$
V	$O(\Delta^4 m^2 n)$ [6]	$O(m^3 e) = O(\Delta m^4)$	$O(n^3 e)$
Any	$O(\Delta^3 m^2(\Delta m + n^3))$	$O(ne(n^2 + e)) = O(\Delta mn(\Delta m + n^2))$	

Table 1: Comparison of our results with the previous results.

Theorem 6. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Suppose G is not convex, i.e. G has one of induced sub-digraphs depicted in Figure 1. Then we have the following.*

1. *Suppose the minimum size forbidden obstruction X of G is of type I or type II. Then X can be found in time $\mathcal{O}(n^2 e)$ [6].*
2. *Suppose the minimum size forbidden obstruction X of G is of type III. Then X can be found in time $\mathcal{O}(ne^2)$.*
3. *Suppose the minimum size forbidden obstruction X of G is of type IV. Then X can be found in time $\mathcal{O}(n^3 e)$.*
4. *Suppose the minimum size forbidden obstruction X of G is of type V. Then X can be found in time $\mathcal{O}(n^3 e)$.*

Theorem 7. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then there is a polynomial time algorithm that decides whether G is convex. If G is not a convex then the algorithms outputs a smallest forbidden subgraph of G. The running time of the algorithm is* $\min\{\mathcal{O}(ne(n^2 + e)), \mathcal{O}(\Delta mn(\Delta m + n^2))\}$.

3 Proofs

Note that without loss of generality we can assume that M does not contain any all-zero columns or rows, as such columns does not affect whether the matrix has the C1P or the forbidden submatrices of M. It follows that $\Delta m \geq n$. We will use this assumption throughout this paper. Also note that the number of edges in $G(M)$ is the same as the number of ones in M, which we denote by e. Note that $e = O(\Delta m)$ and that $e \geq m, n$ (since we assume that there are no all-zero columns or rows in M).

We will use the following auxiliary lemma.

Lemma 2. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then deciding whether G has an induced matching of size two can be done in time $O(e + m + n)$.*

Proof. Order the (white) vertices in W by their degrees: $\deg(w_1) \leq \deg(w_2) \leq \cdots \leq \deg(w_n)$. For every $i = 1, \ldots, n-1$, check if $N(w_i) \setminus N(w_{i+1})$ is non-empty. If for some i, $N(w_i) \setminus N(w_{i+1}) \neq \emptyset$, then also $N(w_{i+1}) \setminus N(w_i) \neq \emptyset$. In this case, we can pick any $a \in N(w_i) \setminus N(w_{i+1})$ and any $b \in N(w_{i+1}) \setminus N(w_i)$, and return edges aw_i, bw_{i+1} as an induced matching of G.

Now, assume that for every i, $N(w_i) \setminus N(w_{i+1}) = \emptyset$, i.e., $N(w_i) \subseteq N(w_{i+1})$. We will show that there is no induced matching of G of size two. Assume for contradiction that aw_i, and bu_j, where $i < j$, is such an induced matching. We have $N(w_i) \subseteq N(w_{i+1}) \subseteq \cdots \subseteq N(w_j)$, i.e., $a \in N(w_j)$, a contradiction. Hence, in this case we can report that there is no such matching.

Vertices of W can be sorted by their degrees in time $O(n+m)$ using a count sort. For each i, checking if $N(w_i) \setminus N(w_{i+1})$ is non-empty can be done in time $O(\deg(w_1))$, hence, the total time spent on checking is $O(\sum_{i=1}^{n-1} \deg(w_i)) = O(e)$. □

3.1 Detection of smallest forbidden subgraphs for each type

We will present four algorithms which find a smallest subgraph of type I, III, IV and V, respectively, each improving the complexity of the best known such algorithm, cf. [3]. For type II, we refer the reader to the $O(ne^3)$ algorithm[1] in [3].

3.1.1 Type I, Proof of Theorem 5 (1)

The proof of Theorem 5 (1) follows from the following lemma.

Lemma 3. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then one can find a minimum size induced subgraph G_I, (type I obstruction) of G in time* $\min\{\mathcal{O}(\Delta e^2), \mathcal{O}(\Delta^3 m^2)\}$ *if there exists one.*

Proof. We apply Algorithm 1 to find a smallest forbidden subgraph of type I in time $O(\Delta e^2)$.

Algorithm 1: Find a smallest G_{I_k} subgraph.

Input : $G = (B, W)$
Output: A smallest subgraph G_{I_k} of G

```
for b ∈ B do
    for x, y ∈ N(b) do
        construct the subgraph G_{b,x,y} of G induced by vertices (B \ (N(x) ∩ N(y))) ∪ (W \ N(b)) ∪ {x, y};
        find a shortest path between x and y in G_{b,x,y};
        if the length of the path is smaller than any observed so far then
            remember b and the vertices of the path;
        end
    end
end
return subgraph of G induced by the remembered set of vertices (if any)
```

Correctness of Algorithm 1. We are looking for induced cycles of length 6 or more in G. For each black vertex b and its two neighbors x, y, we find a shortest induced cycle of

[1]The authors of [3] showed that the complexity of their algorithm is $O(\Delta^4 m^3)$, however, it is easy to check that their algorithm works in time $O(ne^3)$.

length at least 6 containing xb, by as two consecutive edges of such a cycle. Such cycle cannot contain any vertex incident with b other than x and y, and any vertex incident with both x and y other than b. Hence, a shortest such cycle C can be obtained from a shortest $x - y$ path P in $G_{b,x,y}$ by adding two edges xb, yb . This cycle cannot be of length 4, otherwise P would contain a vertex in $N(x) \cap N(y)$. It remains to show that C is induced. Assume that there is a chord uv in C. Since P does not contain $N(b) \setminus \{x, y\}$, we conclude that $u, v \neq b$. Hence, we could use the chord as a shortcut to find a shorter cycle containing edges xb, yb, and hence, a shorter path between x and y in $G_{b,x,y}$, a contradiction.

Complexity of Algorithm 1. We will show that the complexity of Algorithm 1 is $O(\Delta e^2)$. The first loop executes m times and the second $\deg(b)^2$ executed times. Hence, the body of the second loop executes $\sum_{b \in B} \deg(b)^2 = O(\Delta e)$ times. Constructing graph $G_{b,x,y}$ takes time $O(e)$ and finding a shortest path in $G_{b,x,y}$ can be done in time $O(e)$ using the Breadth-first search algorithm. Therefore the running time of Algorithm 1 is $\min\{\mathcal{O}(\Delta e^2), \mathcal{O}(\Delta^3 m^2)\}$.

□

3.1.2 Type III, Proof of Theorem 5 (2)

The proof of Theorem 5 (2) follows from the following lemma.

Lemma 4. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then one can find a minimum size induced subgraph G_{III}, (type III obstruction) in G in time* $\min\{\mathcal{O}(e^3), \mathcal{O}(\Delta^3 m^3)\}$ *if there exists one.*

Proof. We apply Algorithm 2 to find a smallest forbidden subgraph of type III in $O(e^3)$.

Algorithm 2: Find a smallest G_{III_k} subgraph.

Input : $G = (B, W, E)$
Output : A smallest subgraph G_{III_k} of G

```
1  for xw ∈ E, where x ∈ W and w ∈ B do
2      for ya ∈ E, where y ∈ W \ N(w) and a ∈ B \ N(x) do
3          construct the subgraph G_{x,w,y,a} of G induced by vertices (B \ N(x) \ N(y)) ∪ {a} ∪ (N(w) \ {x}) ∪ (W \ N(a));
4          find a shortest path P between a and set W \ N(w) \ N(a) in G_{x,w,y,a} with |P| ≥ 5 (using modified BFS) ;
5          if the P exists and is shorter than any observed so far then
6              remember w, x, y and P ;
7          end
8      end
9  end
10 return subgraph of G induced by the remembered set of vertices (if any)
```

Correctness of Algorithm 2. Let us first verify that the vertices of a shortest path P found in line 4 and w, x, y induce a subgraph of type III. Obviously, x is connected to only w, w is not connected to y and the last vertex z of P . On the other hand, w must be connected to all other white vertices of P, since any such white vertex that is not in $N(w)$ is in $W \setminus N(a)$ and hence, also $W \setminus N(w) \setminus N(a)$, i.e., we would have a shorter path ending at this vertex. Since the path is a shortest path, all black vertices on the path are connected only to its predecessor and successor on the path. In addition a is connected to y and no other black vertex on the path is connected to y since $G_{x,w,y,a}$ does not contain any other neighbors of y. It follows that the vertices w, x, y and the vertices of a shortest path induce

a subgraph of type III. We also note that by just modifiying Breadth First Search to make sure P has at least 5 edges.

Second, consider a smallest subgraph of type III in G. We will show it is considered by the algorithm. Assume the algorithm is in the iteration, where it picked edges xw, ya of this subgraph. Then the rest of the vertices must lie in $G_{x,w,y,a}$: the remaining black vertices are not connected to x, y and the remaining white vertices are either in $N(w) \setminus \{x\}$ and z is $W \setminus N(a)$. These vertices together with a must form a shortest path from a to $W \setminus N(w) \setminus N(a)$ in $G_{x,w,y,a}$, hence, Algorithm 2 finds this subgraph or a subgraph with the same number of vertices.

Complexity of Algorithm 2. We will show that the complexity of Algorithm 2 is $O(e^3)$. The first loop executes e times. The second loop executes $O(e)$ times. Constructing graph $G_{x,w,y,a}$ takes time $O(e)$. Finding a shortest path in G_x can be done in time $O(e)$ using a "modified-breadth-first" search algorithm. Therefore the running time of Algorithm 2 is $\min\{\mathcal{O}(\Delta e^3), \mathcal{O}(\Delta^3 m^3)\}$.

□

3.1.3 Type IV, Proof of Theorem 5 (3)

The proof of Theorem 5(3) follows from the following lemma.

Lemma 5. *Let $G = (B, W)$ be a bipartite graph with m black vertices and n white vertices. Then one can find a minimum size induced subgraph G_{IV}, (type IV obstruction) of G in time $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$ if one exists.*

Proof. We apply Algorithm 3 to find a smallest forbidden subgraph of type IV in $O(m^3 e)$ if one exists.

Algorithm 3: Find a G_{IV} subgraph.

Input : $G = (B, W)$
Output: A subgraph G_{IV} of G

```
for distinct a, b, c, d ∈ B do
    find UX = N(a) \ (N(b) ∪ N(c));
    find VY = N(b) \ (N(a) ∪ N(c));
    find WZ = N(c) \ (N(a) ∪ N(b));
    find U = UX ∩ N(d) and X = UX \ N(d);
    find V = VY ∩ N(d) and Y = VY \ N(d);
    find W = WZ ∩ N(d) and Z = WZ \ N(d);
    if each of the sets X, Y, Z, U, V, W is non-empty then
        pick any x ∈ X, y ∈ Y, z ∈ Z, u ∈ U, v ∈ V, w ∈ W;
        return G[a, b, c, d, x, y, z, u, v, w]
    end
end
return not found
```

Correctness of Algorithm 3. It is easy to see that once a, b, c, d are picked, each of x, y, z, u, v, w has to belong to computed set X, Y, Z, U, V, W, respectively, and that once they are picked from those sets, the returned vertices induce G_{IV}.

Complexity of Algorithm 3. We will show that the complexity of Algorithm 3 is $O(m^3 e)$. The time complexity of the steps inside the loop depends on the degrees of nodes a, b, c, d, i.e., it is $O(\deg(a)+\deg(b)+\deg(c)+\deg(d))$. Hence, the overall complexity is $\sum_{a,b,c,d \in R} O(\deg(a)+$

$\deg(b) + \deg(c) + \deg(d)) = 4\sum_{a,b,c,d \in R} O(\deg(d)) = 4\sum_{a,b,c \in R} O(e) = m^3 e$. Therefore the running time of Algorithm 3 is $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$.

□

3.1.4 Type V, Proof of Theorem 5(4)

The proof of Theorem 5(4) follows from the following lemma.

Lemma 6. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Then one can find a minimum size induced subgraph G_{IV}, (type IV obstruction) of G in time $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$ if one exists.*

Proof. We apply Algorithm 4 to find a smallest forbidden subgraph of type V in $O(m^3 e)$ if one exists.

Algorithm 4: Find a G_{V} subgraph.

Input : $G = (B, W)$
Output: A subgraph G_{V} of G

for *distinct* $a, b, c, d \in B$ **do**
  find $UY = N(b) \cap N(d) \setminus N(c)$;
  find $VZ = N(b) \cap N(c) \setminus N(d)$;
  find $U = UY \cap N(a)$ and $Y = UY \setminus N(a)$;
  find $V = VZ \cap N(a)$ and $Z = VZ \setminus N(a)$;
  find $X = N(a) \setminus (N(b) \cup N(c) \cup N(d))$;
  if *each of the sets* X, Y, Z, U, V *is non-empty* **then**
    pick any $x \in X, y \in Y, z \in Z, u \in U, v \in V$;
    return $G[a, b, c, d, x, y, z, u, v]$
  end
end
return *not found*

Correctness of Algorithm 4. It is easy to see that once a, b, c, d are picked, each of x, y, z, u, v has to belong to computed sets X, Y, Z, U, V, respectively, and that once they are picked from those sets, the returned vertices induce G_{V}.

Complexity of Algorithm 4. The complexity of Algorithm 4 is $O(m^3 e)$. This follows by the same argument as for Algorithm 4. Therefore the running time of Algorithm 3 is $\min\{\mathcal{O}(m^3 e), \mathcal{O}(\Delta m^4)\}$. □

3.2 Detection of a smallest forbidden subgraph

Overall, we will use Dom et al. ([6]) approach to find the smallest forbidden subgraph in G. We will first find a shortest-paths (the sum of the lengths of the three paths) white asteroidal triple A in time $O(n^2 e) = O(\Delta m n^2)$ using the algorithm in [6]. For completeness, we are including this algorithm in this paper as Algorithm 5.

Algorithm 5: Find a smallest white asteroidal triple.

Input : $G = (B, W)$
Output: A smallest white asteroidal triple G

```
for x ∈ W do
    construct a subgraph G_x of G induced by (B \ N(x)) ∪ (W \ {x});
    for y ∈ W \ {x} do
        use BFS to determine the shortest distance d_x(y, z) from y to any other
        z ∈ W \ {x} in G_x;
    end
end
return minarg_{x,y,z∈W}(d_x(y, z) + d_y(x, z) + d_z(x, y))
```

It is easy to check that the complexity of Algorithm 5 is $O(n^2e + n^3) = O(n^2e)$.

A shortest-paths white asteroidal triple A must be in T, but does not need to be a smallest forbidden subgraph. Let ℓ be the sum of the lengths of the three paths of A. If A is of

- type I or II, then it contains ℓ vertices;
- type III, it contains $\ell - 5$ vertices;
- type IV, it contains $10 = \ell - 8$ vertices;
- type V, it contains $9 = \ell - 1$ vertices.

It follows that if one of the smallest forbidden subgraphs is of type I or II, then each shortest-paths asteroidal triple is of type I or II and is a smallest forbidden subgraph. For the remaining cases, we need to determine the smallest forbidden subgraphs of type III, IV and V. However, we only need to find a smallest subgraph of type X if it is a smallest forbidden subgraph. Hence, for types IV and V, if we find during the search that there is a smaller forbidden subgraph of some other type, we can stop searching for this type. For type III, since it has a variable size, we cannot stop searching, however, we can abandon the branch which would yield a larger or even the same size subgraph of type III than we have observed. We will use this in what follows to obtain faster algorithms for types III, IV and V than the ones presented in the previous section.

3.2.1 Type III, Proof of Theorem 6(1)

Lemma 7. *Let $G = (B, W)$ be a bipartite graph with m black vertices and n white vertices. Let O be a minimum size forbidden obstruction in G. Then in $\mathcal{O}(ne^2)$, one can find O if it is of type III or report O is of type I or V.*

Proof. Algorithm 6 guarantees to find a smallest subgraph of type III in time $O(ne^2)$ **if** it is smaller than other types of forbidden subgraphs. If there is a smaller subgraph of type I or there is a smaller of the same size subgraph of type V in G, it either reports that or it could report a subgraph of type III which is not the smallest. It will first determine whether

G_{III_1} is a subgraph of G. If not it continues to the second phase, where it assumes that the smallest subgraph of type III (if it exists) has at least 9 vertices.

Algorithm 6: Find a smallest G_{III_k} subgraph if it is smaller than other types of subgraphs.

Input : $G = (B, W, E)$
Output: A smallest subgraph G_{III_k} of G or report there is a subgraph of other type (I or V) of equal or smaller size

for $w \in B$ **do**
  for $x, u \in N(w)$ **do**
    construct the subgraph $G_{x,w,u}$ of G induced by vertices $(N(u) \setminus N(x)) \cup (W \setminus N(w))$;
    find induced matching of size two using Lemma 2;
    if *induced matching exists* **then**
      return subgraph of G induced by x, w, u and the induced matching (G_{III_1})
    end
  end
end
/* We can now assume that there is no G_{III_1} in G */
set $i_{min} = \infty$;
for $xw \in E$, *where* $x \in W$ *and* $w \in B$ **do**
  find $D = N_2(w) \setminus N(x)$ and $Y = N(D) \setminus N(w)$;
  for $y \in Y$ **do**
    construct the subgraph $G_{x,w,y}$ of G induced by vertices $N(w) \setminus \{x\} \cup \{y\} \cup D$;
    find $D_i = N_i(y)$ in $G_{x,w,y}$, for $i \geq 1$;
    find $Y' = \{y' \in Y : D_1 \setminus N(y') \neq \emptyset\}$ and $D' = D \cap N(Y')$;
    find smallest odd $i \geq 3$ such that $D_i \cap D' \neq \emptyset$ (if possible);
    if *found* **then**
      pick any $d_i \in D_i \cap D'$, any $y' \in Y' \cap N(d_i)$;
      find a path P from d_i to some $d_1 \in D_1$ in $G_{w,w,y}$ of length $i - 1$;
      if $y'd_1 \notin E$ *and* $i < i_{min}$ **then**
        set i_{min} to i;
        remember x, w, y, y' and vertices of P;
      end
    end
  end
end
if $i_{min} = \infty$ **then**
  return subgraph of type III not found or there is a subgraph of type I or V of the size at most the size of the smallest type III subgraph
else
  return *subgraph of G induced by remembered set of vertices*
end

Correctness of Algorithm 6. It is easy to check that the first phase of the algorithm finds G_{III_1} subgraph if it exists in G. Assume that G_{III_1} is not an induced subgraph of G., i.e., that a smallest subgraph of type III (if it exists) has at least 9 vertices. The algorithm continues to the second phase.

First, assume that i is not found, i.e., for all odd $i \geq 3$, $D_i \cap D' = \emptyset$. This implies that any path starting at y in $G_{x,w,y}$ cannot be extended with a white vertex y' that is not adjacent to w and not adjacent to the second vertex $d_1 \in D_1$ of this path. Hence, the algorithm correctly continues with examining another selection of vertices x, w, y. Assume that i was found. Now, assume that G does not contain edge $y'd_1$. Let us verify that vertices x, w, y, y' and the vertices of P induce $G_{\mathrm{III}_{(i-1)/2}}$. It is clear that x is connected only to w and w only to white vertices on P except the first vertex y. By the construction, each vertex on P can be adjacent only to its predecessor or successor on P. Since $i > 2$ is the smallest odd integer such that $D_i \cap D' \neq \emptyset$, y' is not adjacent to any black vertex on P other than the last one. Hence, the vertices induce a subgraph of type III. Finally, assume that $y'd_1 \in E$. If $i \geq 5$ then the vertices of P without y and y' induce a cycle of length $i+1$, i.e., a subgraph $G_{\mathrm{I}_{(i-3)/2}}$, which is smaller than a subgraph of type III we could get for this selection

of x, w, y (by choosing a different d_i, y' or path P, or searching for another odd i such that $D_i \cap D' \neq \emptyset$). If $i = 3$, consider $d'_1 \in D_1$ that is not adjacent to y' and let $P = y, d_1, u, d_3$. If d'_1 is adjacent to u, vertices $x, w, u, d'_1, y, d_3, y'$ induce G_{III_1}, a contradiction. Hence, assume $d'_1 u \notin E$. Since $d'_1 \in D \subseteq N_2(w)$, there exists $u' \in N(w)$ adjacent to d'_1. If $d_1 u' \in E$, then vertices $x, w, u, u', d_1, d'_1, d_3, y, y'$ induce G_{V}. Otherwise, vertices w, u, d_1, y, d'_1, u' induce a cycle of length 6. In any case, there exists a subgraph of other type of size equal or smaller than it would be possible to find for this choice of x, w, y, hence, the algorithm correctly moves to the next choice.

Complexity of Algorithm 6. We will show that the complexity of Algorithm 6 is $O(ne^2) = O(\Delta^2 m^2)$. The body of the loop in lines 2–7 will execute $O(\Delta e)$ times and each step of the body takes $O(e)$ time. Hence, the complexity of the first phase is $O(\Delta e^2) = O(ne^2)$. The main loop of the second phase will execute $O(e)$ times. Determining D and Y takes time $O(e)$. The nested loop in lines 13–26 will execute $O(n)$ times. Each step of the body of this loop will take time $O(e)$. Hence, the complexity of the second phase is $O(ne^2)$.

□

3.2.2 Type IV, Proof of Theorem 6(2)

In this subsection we give two algorithms. Each of them finds a minimum sized induced subgraph G_{IV} of G or they report the minimum size forbidden subgraph of G is of type I or III.

Algorithm 7: Find a G_{IV} subgraph or report that there is a smaller subgraph of type I or III.

Input : $G = (B, W)$
Output: A subgraph G_{IV} of G or report that G_{IV} is not a smallest subgraph

```
1  for distinct x, y, z ∈ W do
2      find A = N(x) \ (N(y) ∪ N(z));
3      find B = N(y) \ (N(x) ∪ N(z));
4      find C = N(z) \ (N(x) ∪ N(y));
5      find D = W \ (N(x) ∪ N(y) ∪ N(z));
6      find U = N(A) \ {x, y, z};
7      find V = N(B) \ {x, y, z};
8      find W = N(C) \ {x, y, z};
9      if all sets A, B, C, D, U, V, W are non-empty then
10         for d ∈ D do
11             if there exists distinct u ∈ U ∩ N(d), v ∈ V ∩ N(d) and w ∈ W ∩ N(d) then
12                 find a ∈ A ∩ N(u), b ∈ B ∩ N(v) and c ∈ C ∩ N(w);
13                 if none of the edges av, aw, bu, bw, cu, cv exists then
14                     return G[x, y, z, u, v, w, a, b, c, d] = G_IV
15                 else
16                     return there is a smaller subgraph of type I or III
17                 end
18             end
19         end
20     end
21 end
22 return not found
```

Lemma 8. *Let $G = (B, W, E)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Let O be a minimum size forbidden obstruction in G. Then one can find O in time $\min\{\mathcal{O}(n^3 e), \mathcal{O}(\Delta m n^3)\}$ if O is of type IV otherwise reports O is of type I or III.*

Proof. Algorithm 7 finds the subgraph G_{IV} in time $O(n^3e)$, if it exists and if it is a smallest forbidden subgraph. If there is a smaller forbidden subgraph of type I or III, it might find an instance of G_{IV} or it might report that there is a smaller forbidden subgraph instead.

Correctness of Algorithm 7. Correctness of the algorithm follows by the following claim.

Claim 1. *Consider a subgraph G' of G induced by vertices $x, y, z, u, v, w, a, b, c, d$ that contains edges $xa, yb, zc, au, bv, cw, ud, vd, wd$ and does not contain edges xd, yd, zd.*

Then either G' is an instance of G_{IV} or G' contains either G_{I_1}, G_{III_1} or G_{III_2} as an induced subgraph.

Proof. We will use the following two partial maps: $R(x) = a$, $R(y) = b$, $R(z) = c$, $R(a) = u$, $R(b) = v$ and $R(c) = w$, and $L = R^{-1}$.

If none of the edges in $E' = \{av, aw, bu, bw, cu, cv\}$ is present in G', then G' is isomorphic to G_{IV}.

If exactly one edge μ in E' is present in G' then we have an induced subgraph G_{III_1} centered at the vertex $r = \mu \cap \{u, v, w\}$. In particular, vertices $d, r, L(r), L(L(r)), \ell, L(\ell), z$, where $\ell = \mu \cap \{a, b, c\}$ and $z \in \{u, v, w\} \setminus \{r, R(\ell)\}$, induce G_{III_1}.

We can assume that there are at least two edges in E' are present in G'. We will distinguish two cases. Either (i) there exist two edges μ, μ' of G' in E' such that $\mu \cap \mu' \neq \emptyset$, or (ii) for each pair of such edges $\mu \cap \mu' = \emptyset$.

First, consider case (i) and suppose $\mu \cap \mu' \neq \emptyset$. Depending on whether the intersection lies in $\{a, b, c\}$ or $\{u, v, w\}$, we have two cases:

1. $\mu \cap \mu' \in \{a, b, c\}$ ("edges joing on the left"), then vertices $V(G') \setminus \{\mu \cap \mu'\}$ induce G_{III_2};
2. $\mu \cap \mu' \in \{u, v, w\}$ ("edges joing on the right"), then vertices $x, y, z, a, b, c, \mu \cap \mu'$ induce G_{III_1}.

Now, consider case (ii). Note the number of edges in E' that are edges of G' is at most three. We will consider two cases depending on the number of such edges:

1. $|E' \cap E(G')| = 2$: Without loss of generality we can assume that $\mu \cap \{a, b, c\} = L(\mu' \cap \{u, v, w\})$ for $\mu, \mu' \in E'$ present in G'. Then the same collection of vertices as in the case of one edge μ induces G_{III_1}, since one end of μ' lies outside of this collection.
2. $|E' \cap E(G')| = 3$: Then the vertices a, b, c, u, v, w induce C_6, i.e., G_{I_1}.

□

Complexity of Algorithm 7. We will show that the complexity of Algorithm 7 is $O(n^3e) = O(\Delta mn^3)$. The first loop executes $O(n^3)$ times, determining A, B, C, D takes time $O(m)$, determining sets U, V, W time $O(e)$. The loop for $d \in D$ is executed $O(m)$ times and each execution takes time $O(deg(d))$, i.e., the total time spent in this loop is $\sum_{d \in D} O(\deg(d)) = O(e)$.

□

In what follow we present a different algorithm to find a minimum size obstruction of type IV.

Algorithm 8: Find a G_{IV} subgraph or report that there is a smaller subgraph of type I or III.

Input : $G = (B, W)$
Output: A subgraph G_{IV} of G or report that G_{IV} is not a smallest subgraph

for $x, y \in W$ **do**
  find and store $N(N(x) \setminus N(y)) \setminus \{x\}$;
end
for $d \in B$ **do**
  for *distinct* $u, v, w \in N(d)$ **do**
    find $X = N(u) \setminus (N(v) \cup N(w))$;
    find $Y = N(v) \setminus (N(u) \cup N(w))$;
    find $Z = N(w) \setminus (N(u) \cup N(v))$;
    if $N(u) \cap N(v) \setminus N(w) \neq \emptyset$ *and* $N(u) \cap N(w) \setminus N(v) \neq \emptyset$ *and* $N(v) \cap N(w) \setminus N(u) \neq \emptyset$ **then**
      return *there is a smaller subgraph of type I* (G_{I_1})
    end
    /* W.l.o.g. assume that $N(u) \cap N(w) \setminus N(v) = \emptyset$ */
    pick any $x \in N(N(u) \setminus N(v)) \setminus N(d)$ if possible;
    pick any $y \in N((N(v) \setminus N(w)) \setminus N(d)$ if possible;
    pick any $z \in N(N(w) \setminus N(v)) \setminus N(d)$ if possible;
    if x, y, z *exist* **then**
      pick any $a \in X \cap N(x)$;
      pick any $b \in Y \cap N(y)$ if possible;
      if b *does not exists* **then**
        return *there is a smaller subgraph of type III* (G_{III_1})
      end
      pick any $c \in Z \cap N(z)$;
      if *any of the edges* av, aw, bu, bw, cu, cv *exists* **then**
        return *there is a smaller subgraph of type I* (G_{I_1})
      end
      return $G[x, y, z, u, v, w, a, b, c, d] = G_{\mathrm{IV}}$
    end
  end
end
return *not found*

Lemma 9. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Let O be a minimum size forbidden obstruction in G. Then one can find O in time $\mathcal{O}((n^2 + \Delta^3 + \Delta^2 m)e)$ if O is of type IV otherwise reports O is of type I or III.*

Proof. Correctness of Algorithm 8. Note that in order to save time the algorithm does not try all possibilities for x, y, z and a, b, c, but rather picks one choice (if possible). After this choice is made, it will either find G_{IV} or it will report that there is a smaller forbidden subgraph. Let us now verify that decisions algorithm makes are correct:

- First, assume that the algorithm stops in line 10. Let $a' \in N(u) \cap N(v) \setminus N(w)$, $b' \in N(u) \cap N(w) \setminus N(v)$ and $c' \in N(v) \cap N(w) \setminus N(u)$. The vertices u, a', v, b', w, c' induce $C_6 = G_{\mathrm{I}_1}$, i.e., there is a smaller induced subgraph of type I.

- Otherwise, we will argue that in line 16, $X \cap N(x)$ is always non-empty, i.e, there is always a choice for a. Since $x \in N(N(u) \setminus N(v)) \setminus N(d)$, there is $a \in N(u) \setminus N(v)$ such that a and x are adjacent. If we have also $a \in N(w)$, then $a \in N(u) \cap N(w) \setminus N(v)$, which is empty. Hence, $a \notin N(w)$, and thus, $a \in X$. The similar argument implies that in line 21, $Z \cap N(z) \neq \emptyset$.

- Next, assume that the algorithm stops in line 25. Since $y \in N(N(v) \setminus N(w)) \setminus N(d)$, there is $b' \in N(v) \setminus N(w)$ such that b' and y are adjacent. Since there is no $b \in= N(v) \setminus (N(u) \cup N(w))$ such that b and y are adjacent, we have $b' \in N(v) \cap N(u) \setminus N(w)$. It follows that the vertices y, d, w, a, x, b', y induce G_{III_1}.

- If the algorithm stops in line 23, at least one of the edges av, aw, bu, bw, cu, cv is present. Consider any such edge and follow its endpoints through set $X \cup Y \cup Z \cup \{u, v, w\}$ back to d. We have an induced $C_6 = G_{\mathrm{I}_1}$.

- Finally, it is easy to check that if the algorithm outputs an induced subgraph in line 25, it is G_{IV}.

On the other hand, if G_{IV} is a smallest forbidden subgraph of G, then the algorithm cannot finish in lines 10, 19 and 23, and hence, it will eventually output G_{IV} in line 25.

Complexity of Algorithm 8. We will show that the complexity of Algorithm 8 is $O((n^2 + \Delta^3 + \Delta^2 m)e) = O((n^2 + \Delta^3 + \Delta^2 m)\Delta m)$. The loop in lines 1–3 executes $O(n^2)$ times and each run takes $O(e)$ time, i.e., the total time spent on computing second neighborhoods $N_2(x)_y$ is $O(n^2 e)$. The loop in lines 4–28 will execute $O(m)$ times and the loop in lines 5–27 $O(\deg(d)^3)$ times. Finding X, Y, Z as well as testing condition in line 9 can be done in time $O(m)$. Picking x, y, z can be done in time $O(\Delta)$: for example, to pick x, we can enumerate through vertices in $N(N(u) \setminus N(v))$ and check each if it is not adjacent to d. Since $|N(d)| \leq \Delta$, the x will be picked (or it will be determined that no such x exists) in at most $\Delta + 1$ steps. Picking a, b, c can be done in time $O(m)$ and testing for presence of edges in line 22 takes a constant time. Hence, the complexity of the loop in lines 4–28 is $\sum_{d \in B} \deg(d)^3 (O(m + \Delta)) = O(\Delta^2 e(m + \Delta))$.

□

3.2.3 Type V

In this subsection we give two algorithms, Algorithm 9 and Algorithm 10. Each of them finds a minimum sized induced subgraph G_V of G or they report the minimum size forbidden subgraph of G is of type I or III. We will show that the complexity of Algorithm 9 is $O(n^3e)$ and the complexity of Algorithm 10 is $O(me^2)$. Hence, the subgraph of type V (if it is a smallest forbidden subgraph) can be found in time $\min\{\mathcal{O}(n^3e), \mathcal{O}(me^2)\}$.

Lemma 10. *Let $G = (B, W)$ be a bipartite graph with m black vertices and n white vertices. Let O be a minimum size forbidden obstruction in G. Then one can find O in time $\mathcal{O}(n^3e)$ if O is of type V otherwise reports O is of type I or III.*

Algorithm 9: Find a G_{V} subgraph or report that there is a smaller subgraph of type I or III.

Input : $G = (B, W)$
Output: A subgraph G_{V} of G or report that G_{V} is not a smallest subgraph

```
1  for distinct x, y, z ∈ W do
2      find X = N(x) \ (N(y) ∪ N(z));
3      find Y = N(y) \ (N(x) ∪ N(z));
4      find Z = N(z) \ (N(x) ∪ N(y));
5      find U = (N(y) ∩ N(z)) \ N(x);
6      pick any u ∈ N(X) ∩ N(Y) ∩ N(U) if possible;
7      pick any v ∈ N(X) ∩ N(Z) ∩ N(U) if possible;
8      if u and v has been picked then
9          if u ∈ N(Z) or v ∈ N(Y) then
10             return there is a smaller subgraph of type III (G_III1)
11         end
12         find X' = X ∩ N(u) ∩ N(v) and U' = U ∩ N(u) ∩ N(v);
13         if X' = ∅ or U' = ∅ then
14             return there is a smaller subgraph of type I (G_I1 or G_I2)
15         end
16         pick any a ∈ X', b ∈ Y ∩ N(u), c ∈ Z ∩ N(v) and d ∈ U';
17         return G[x, y, z, u, v, a, b, c, d]
18     end
19 end
20 return not found
```

Proof. Correctness of Algorithm 9. The algorithm is able to reduce time complexity by avoiding trying all possible choices for u, v and a, b, c, d, but rather picking one choice (if possible), and then either finding G_{V} or a smaller forbidden subgraph. Let us verify that decisions algorithm makes are correct:

- First, assume that the algorithm stops in line 10. Then there exists $w \in N(X) \cap N(Y) \cap N(Z) \cap N(U)$ (either u or v). Then there exists $a \in X \cap N(w)$, $b \in Y \cap N(w)$ and $c \in Z \cap N(w)$. Vertices x, y, z, a, b, c, w induce G_{III_1}.

- Assume that the algorithm stops in line 14. If $X' = \emptyset$ and $U' = \emptyset$, there exists $a \in X \cap N(u)$, $a' \in X \cap N(v)$, $d \in U \cap N(u)$ and $d' \in U \cap N(v)$. Note that $a \neq a'$, $d \neq d'$, $a, d \notin N(v)$ and $a', d' \notin N(u)$. It is easy to check that vertices x, a, u, d, y, d', v, a' induce C_8. Similarly, if either $X' = \emptyset$ or $U' = \emptyset$, we can find vertices that induce C_6.

- Finally, it is easy to check that if the algorithm outputs an induced subgraph in line 17, it is G_{V}.

On the other hand, if G_{V} is a smallest forbidden subgraph of G, then the algorithm cannot finish in lines 10 and 14, and hence, it will eventually output G_{IV} in line 17.

Complexity of Algorithm 9. We will show that the complexity of Algorithm 9 is $O(n^3 e) = O(\Delta m n^3)$. The first loop executes $O(n^3)$ times, determining X, Y, Z, U takes time $O(m)$, picking u, v time $O(e)$, picking a, b, c, d time $O(m)$. Hence, the total time used by the algorithm is $O(n^3(O(m) + O(e))) = O(n^3 e)$.

□

Lemma 11. *Let $G = (B, W)$ be a bipartite graph with m black vertices, n white vertices, and e edges. Let O be a minimum size forbidden obstruction in G. Then one can find O in time $\mathcal{O}(me^2)$ if O is of type V otherwise reports O is of type I or III.*

Algorithm 10: Find a G_{V} subgraph or report that there is a smaller subgraph of type I, II or III.

Input : $G = (B, W)$
Output: A subgraph G_{V} of G or report that G_{V} is not a smallest subgraph

```
1  for distinct a, d ∈ B do
2      find X = N(a) \ N(d);
3      find UV = N(a) ∩ N(d);
4      find YZ = N(d) \ N(a);
5      for x ∈ X do
6          find BC = N(YZ) ∩ N(UV) \ {d} \ N(x);
7          pick any induced matching yb and zc such that y, z ∈ YZ and b, c ∈ BC if
           possible;
8          if edges yb, zc exist then
9              if both edges bx, cx exist then
10                 return there is a smaller subgraph of type I (G_{I_1})
11             end
12             if exactly one of bx, cx exists then
13                 return there is a smaller subgraph of type I or II (G_{I_1} or G_{II_1})
14             end
15             pick any u ∈ UV ∩ N(b);
16             if uc is an edge then
17                 return there is a smaller subgraph of type III (G_{III_1})
18             end
19             pick any v ∈ UV ∩ N(c);
20             return G[x, y, z, u, v, a, b, c, d]
21         end
22     end
23 end
24 return not found
```

Proof. Correctness of Algorithm 10. Let us verify that the decisions the algorithm makes are correct:

- First, assume that the algorithm stops in line 10, then vertices d, y, b, x, c, z induce $C_6 = G_{\mathrm{I}_1}$.
- Suppose the algorithm stops in line 13. Without loss of generality assume that $\{b, x\}$ is an edge and c and x are not connected in G. There must exists a $v \in UV \cap N(c)$. If b and v are noc connected, then vertices d, y, b, x, a, v induce $C_6 = G_{\mathrm{I}_1}$. If there is edge bv in G, vertices y, b, d, x, v, z, a, c induce G_{II_1}.
- Suppose the algorithm stops in line 17, then vertices u, a, x, b, y, c, z induce G_{III_1}.
- Finaly, it is easy to check that if the algorithms outputs an induced subgraph in line 20, it is G_{V}.

On the other hand, if G_{V} is a smallest forbidden subgraph, the algorithm cannot finish in lines 10, 13 and 17, and hence, it will eventually output G_{V} in line 20.

Complexity of Algorithm 10. We will show that the complexity of the algorithm if $O(me^2) = O(\Delta^2 m^3)$. The main loop will execute $O(m^2)$ times, finding X, UV, YZ will take time $O(\deg(a) + \deg(d))$. The second loop will execute $O(\deg(a))$ times. As before, the running time of the algorithm is $O(meT)$, where T is the time spend inside the second loop. Finding BC takes time $O(e)$. The following lemma shows that picking induced matching of size two can be done in time $O(e)$. The rest of the body of the second loop takes time $O(\Delta)$, hence, the complexity of the algorithm is $O(me^2)$.

□

3.2.4 Main algorithm, Proof of Theorem 7

The proof follows from the Algorithm 11 and the argument about it correctness and its complexity. Algorithm 11 finds a smallest forbidden subgraph using the three algorithms described above.

Algorithm 11: Find a smallest forbidden Tucker subgraph.

Input : $G = (B, W)$
Output: A smallest forbidden subgraph of G

```
1  find a smallest white asteroidal triple A using Lemma 1;
2  let ℓ be the sum of the lengths of three paths of A;
3  find a smallest subgraph of types III, IV and V (using the procedures described above);
4  let s_III, s_IV, s_V be the sizes of these subgraphs (or ∞ if not found), respectively;
5  if ℓ = min{ℓ, s_III, s_IV, s_V} then
6  |  return A
7  else
8  |  let s_X = min{ℓ, s_III, s_IV, s_V};
9  |  return the smallest subgraph of type X
10 end
```

To verify the correctness of Algorithm 11, first consider that one of the smallest forbidden subgraphs of G is of type I or II. By the above argument, asteroidal triple A is of type I or II with size ℓ, and since it is a smallest forbidden subgraph, we have $\ell = \min\{\ell, s_{\mathrm{III}}, s_{\mathrm{IV}}, s_{\mathrm{V}}\}$. Hence, the algorithm correctly outputs one of the smallest forbidden subgraphs. Second, assume that all smallest forbidden subgraphs of G are of type III, IV and V. Let $s = \min\{s_{\mathrm{III}}, s_{\mathrm{IV}}, s_{\mathrm{V}}\}$. If A is of type I or II, then the size of A is ℓ, and hence, $\ell > s$ and

$s_X = \min\{\ell, s_{\mathrm{III}}, s_{\mathrm{IV}}, s_{\mathrm{V}}\}$. If A is of type III, IV or V, then $\ell \geq s + 1$, and hence again $s_X = \min\{\ell, s_{\mathrm{III}}, s_{\mathrm{IV}}, s_{\mathrm{V}}\}$. It follows that Algorithm 11 correctly outputs one of the smallest forbidden subgraphs.

It follows from Algorithm 11 that we do not need a special detection algorithms for type I and II forbidden subgraphs. However, in some applications, there might be a need to determine a smallest forbidden subgraph of each type. Therefore, we present such algorithms for these two types of forbidden subgraphs as well.

References

[1] Zaky Adam, Monique Turmel, Claude Lemieux, David Sankoff: *Common Intervals and Symmetric Difference in a Model-Free Phylogenomics, with an Application to Streptophyte Evolution*. Journal of Computational Biology 14(4): 436-445 (2007)

[2] Farid Alizadeh, Richard M. Karp, Lee Aaron Newberg, Deborah K. Weisser: *Physical Mapping of Chromosomes: A Combinatorial Problem in Molecular Biology*. Algorithmica 13(1/2): 52-76 (1995)

[3] Guillaume Blin and Romeo Rizzi and Stéphane Vialette: *A Faster Algorithm for Finding Minimum Tucker Submatrices*. J. Theory Comput. Syst. (51) 270-281 (2012)

[4] Kellogg S. Booth and George S. Lueker: *Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms*. J. Comput. Syst. Sci. 13(3): 335-379 (1976)

[5] Cedric Chauve, Eric Tannier: *A Methodological Framework for the Reconstruction of Contiguous Regions of Ancestral Genomes and Its Application to Mammalian Genomes*. PLoS Computational Biology 4(11) (2008)

[6] Michael Dom and Jiong Guo and Rolf Niedermeier : *Approximation and fixed-parameter algorithms for consecutive ones submatrix problems* J.Computer and System Sciences 76 (3-4) 204-221 (2010)

[7] M.Habib, Ross M. McConnell, Christophe Paul, Laurent Viennot: *Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing*. Theor. Comput. Sci. 234(1-2): 59-84 (2000)

[8] Wen-Lian Hsu: *A Simple Test for the Consecutive Ones Property*. J. Algorithms 43(1): 1-16 (2002)

[9] Wei-Fu Lu, Wen-Lian Hsu: *A Test for the Consecutive Ones Property on Noisy Data - Application to Physical Mapping and Sequence Assembly*. Journal of Computational Biology 10(5): 709-735 (2003)

[10] Nathan Lindzey and Ross M. McConnell : *On Finding Tucker Submatrices and Lekkerkerker-Boland Subgraphs*. WG 2013.

[11] Jian Ma, Louxin Zhang, Bernard B. Suh, Brian J. Raney, Richard C. Burhans, W. James Kent, Mathieu Blanchette, David Haussler, and Webb Miller1 : *Reconstructing contiguous regions of an ancestral genome*. GenomeRes 16(12) 1557–1565 (2006)

[12] Ross M. McConnell: *A certifying algorithm for the consecutive-ones property.* SODA 2004: 768-777

[13] Joao Meidanis, Oscar Porto, Guilherme P. Telles: *On the Consecutive Ones Property.* Discrete Applied Mathematics 88(1-3): 325-354 (1998)

[14] A. C. Tucker: *A structure theorem for the consecutive 1's property.* J. of Comb. Theory, Series B 12 :153-162 (1972)