Bounded Degree Spanning Trees[⋆]

(Extended abstract)

Artur Czumaj and Willy-B. Strothmann

Heinz Nixdorf Institute and Department of Computer Science
University of Paderborn, D-33095 Paderborn, Germany
{artur,willy}@uni-paderborn.de

Abstract. Given a connected graph G, let a Δ_T-spanning tree of G be a spanning tree of G of maximum degree bounded by Δ_T. It is well known that for each $\Delta_T \geq 2$ the problem of deciding whether a connected graph has a Δ_T-spanning tree is $\mathcal{NP}$-complete. In this paper we investigate this problem when additionally connectivity and maximum degree of the graph are given. A complete characterization of this problem for 2- and 3-connected graphs, for planar graphs, and for $\Delta_T = 2$ is provided.

Our first result is that given a biconnected graph of maximum degree $2\Delta_T - 2$, we can find its Δ_T-spanning tree in time $O(m + n^{3/2})$. For graphs of higher connectivity we design a polynomial-time algorithm that finds a Δ_T-spanning tree in any k-connected graph of maximum degree $k(\Delta_T - 2) + 2$. On the other hand, we prove that deciding whether a k-connected graph of maximum degree $k(\Delta_T - 2) + 3$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete, provided $k \leq 3$. For arbitrary $k \geq 3$ we show that verifying whether a k-connected graph of maximum degree $k(\Delta_T - 1)$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete. In particular, we prove that the Hamiltonian path (cycle) problem is $\mathcal{NP}$-complete for k-connected k-regular graphs, if $k > 2$. This extends the well known result for $k = 3$ and fully characterizes the case $\Delta_T = 2$.

For planar graphs it is $\mathcal{NP}$-complete to decide whether a k-connected planar graph of maximum degree Δ_G has a Δ_T-spanning tree for $k = 1$ and $\Delta_G > \Delta_T \geq 2$, for $k = 2$ and $\Delta_G > 2(\Delta_T - 1) \geq 2$, and for $k = 3$ and $\Delta_G > \Delta_T = 2$. On the other hand, we show how to find in polynomial (linear or almost linear) time a Δ_T-spanning tree for all other parameters of k, Δ_G, and Δ_T.

1 Introduction

Various problems of constructing spanning trees, or generally spanning subgraphs, that satisfy given constraints have been studied before extensively [1,4,5,10,13–16,19]. Such problems often arise in designing communication networks for the purpose of broadcasting or fault tolerance (e.g., see discussion in [4]). It has been shown that in most of the cases even simple constraints make the problem $\mathcal{NP}$-hard [8,20]. For example, it is well known that for any $\Delta_T \geq 2$ the problem of testing whether a graph has a spanning tree of maximum degree bounded by Δ_T is $\mathcal{NP}$-complete [8].

[⋆] Partially supported by EU ESPRIT Long Term Research Project 20244 (ALCOM-IT), DFG Leibniz Grant Me872/6-1, and DFG Project Me872/7-1.

Let Δ_T be an arbitrary integer greater than 1. A Δ_T*-spanning tree* of G is a spanning tree of G of maximum degree bounded by Δ_T. When can we precisely say that a given graph has a Δ_T-spanning tree, and if it has one, then how efficiently it can be found? Even in the most basic case $\Delta_T = 2$, that is, of verifying whether a graph has a Hamiltonian path, only very few results are known. For example, it is known that when a graph is sufficiently dense or its degree sequence satisfies certain conditions, then it has a Hamiltonian path and one such path can be constructed in polynomial time (see e.g. [3]). In the case of planar graphs, Tutte showed that every 4-connected planar graph has a Hamiltonian cycle and Chiba and Nishizeki [6] provided a linear-time algorithm for the construction. On the opposite side, Garey et al. [9] proved that it is $\mathcal{NP}$-complete to decide whether a 3-connected 3-regular planar graph has a Hamiltonian path.

Even less is known for larger values of Δ_T. Neumann-Lara and Rivera-Campo [19] characterized the values of Δ_T for spanning trees of k-connected graphs as a function of its independence number, and Caro et al. [5] as a function of its minimum degree. Both these results are mainly interesting for dense graphs and are far from being tight for sparse graphs. Barnette [1] showed that every 3-connected planar graph has a 3-spanning tree. Perhaps the most general result was obtained by Fürer and Raghavachari [7] and Win [25]. They showed that the optimal value of Δ_T is approximated within an additive constant term by the inverse of the toughness of the graph. We notice however, that from the algorithmic point of view this result would be not satisfactory, because it is $\mathcal{NP}$-hard to determine the toughness of graphs [2]. Nevertheless, Fürer and Raghavachari [7] were able to estimate the toughness of graphs and designed a polynomial-time algorithm that finds a spanning tree whose maximum degree is within one of optimal.

1.1 New Results

In this paper we investigate the problem of testing whether a graph has a Δ_T-spanning tree when connectivity and maximum degree of the graph are given as parameters. Let G be a k-connected graph of maximum degree $\Delta_G \geq k$ with n vertices and m edges.

We show that increasing the connectivity of graphs provides much weaker conditions on the maximum degree of the graph to ensure the existence and efficient finding of spanning trees of low degree. Our first algorithm is designed for biconnected graphs and runs in $O(m + n^{3/2})$ time. It finds a spanning tree T in which the degree of every vertex v is roughly halved; more precisely, $d_T(v) \leq \lceil d_G(v)/2 \rceil + 1$. In particular, if a graph is of maximum degree at most $2\Delta_T - 2$, then the algorithm finds a Δ_T-spanning tree of G. The second algorithm runs in time $O(n^2 \cdot k \cdot \alpha(n, n) \cdot \log n)$ and finds a Δ_T-spanning tree of any k-connected graph of maximum degree bounded by $k(\Delta_T - 2) + 2$. These results improve significantly upon the trivial lower degree bounds for $k \geq 2$ and/or $\Delta_T \geq 3$.

Then we prove that the Hamiltonian path problem and the Hamiltonian cycle problem are $\mathcal{NP}$-complete for k-connected k-regular graphs, $k \geq 3$. This extends the well-known result for 3-connected graphs [9] and fully characterizes the complexity of the problem for $\Delta_T = 2$ and $k \geq 3$. We can generalize this bound to $\Delta_T > 2$ and prove that (unless $\mathcal{P} = \mathcal{NP}$) our algorithmic results are in a sense best possible for $k \leq 3$ and $\Delta_T > 2$. For that we first construct k-connected graphs of maximum degree $k(\Delta_T - 1)$

General graphs					
$k = 2$		$k = 3$		$k \geq 4$	
$\Delta_G \leq 2\Delta_T - 2$	$\Delta_G > 2\Delta_T - 2$	$\Delta_G \leq 3\Delta_T - 4$	$\Delta_G > 3\Delta_T - 4$	$\Delta_G \leq k(\Delta_T - 2) + 2$	$\Delta_G \geq k(\Delta_T - 1)$
$O(m + n^{3/2})$	$\mathcal{NPC}$	$O(n^2\alpha(n, n)\log n)$	$\mathcal{NPC}$	$O(n^2 k\alpha(n, n)\log n)$	$\mathcal{NPC}$

Planar graphs					
$k = 1$	$k = 2$		$k = 3$		$k \in \{4, 5\}$
$\Delta_G > \Delta_T$	$\Delta_G \leq 2\Delta_T - 2$	$\Delta_G > 2\Delta_T - 2$	$\Delta_T = 2$	$\Delta_T \geq 3$	$\Delta_T \geq 2$
$\mathcal{NPC}$[8]	$O(n \log n)$	$\mathcal{NPC}$	$\mathcal{NPC}$[9]	$O(n)$ [24]	$O(n)$ [6]

Table 1. Summary of results. $\mathcal{NPC}$means that the problem is $\mathcal{NP}$-complete.

without Δ_T-spanning trees for $k \geq 3$ and $\Delta_T > 2$. Then we prove that verifying whether a k-connected graph of maximum degree $k(\Delta_T - 1)$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete for $k \geq 3$. Finally, we extend these results and show that it is $\mathcal{NP}$-complete to decide whether a 2-connected (planar) graph of maximum degree $2\Delta_T - 1$ has a Δ_T-spanning tree. These results establish a complete characterization of the Δ_T-spanning tree problem for $k \leq 3$ in general graphs.

Our results can be applied to planar graphs. It is known that every 4-connected planar graph has a Hamiltonian path and that such a path can be found in linear time [6]. Barnette [1] showed that every 3-connected planar graph has a 3-spanning tree and one can also find such a tree in linear time [24]. On the other hand, Garey et al. [9] proved that it is $\mathcal{NP}$-complete to decide whether a 3-regular 3-connected planar graph has a Hamiltonian path. One can easily extend this result to show that it is $\mathcal{NP}$-complete to verify whether a connected planar graph of maximum degree $\Delta_T + 1$ has a Δ_T-spanning tree.

In this paper we fill the remaining gap and characterize biconnected planar graphs. Our result for arbitrary graphs implies that every 2-connected planar graph of maximum degree at most $2(\Delta_T - 1)$ has a Δ_T-spanning tree. In the case of planar 2-connected graphs we can design an algorithm that finds a Δ_T-spanning tree in time $O(n \log n)$. Since we can prove that it is $\mathcal{NP}$-complete to decide whether a 2-connected planar graph of maximum degree $2\Delta_T - 1$ has a Δ_T-spanning tree, this result establishes a complete characterization of the Δ_T-spanning tree problem for k-connected planar graphs of maximum degree Δ_G.

Table 1 summarizes the results (it assumes that $\Delta_G > \Delta_T \geq 2$).

Organization of the paper Section 2 provides basic terminology. Then in Sect. 3 an algorithm for finding a Δ_T-spanning tree in a 2-connected graph of maximum degree at most $2(\Delta_T - 1)$ is presented. Section 4 contains a polynomial-time algorithm that finds a Δ_T-spanning tree of k-connected graphs with maximum degree $\Delta_G \leq k(\Delta_T - 2) + 2$ for arbitrary $k \geq 2$. In Sect. 5 we prove that verifying whether a k-connected graph of maximum degree $k(\Delta_T - 1)$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete for every $k \geq 3$, $\Delta_T \geq 2$. In Sect. 6 we provide a characterization of the Δ_T-spanning tree problem in planar graphs.

Some technical details are omitted here but they will appear in the final version.

2 Basic Notation

We use [3] for basic terminology and notation not defined here. The set of vertices adjacent to a vertex v in a graph G is denoted by $\Gamma_G(v)$ and its size by $d_G(v) := |\Gamma(v)|$. For an arbitrary connected graph H let $\mathbb{BCT}(H)$ denote its *block-cutvertex tree* whose vertices correspond to the *articulation points* of H and to the *blocks* (maximal 2-connected subgraphs) of H, and there is an edge between an articulation point v and a block B iff v is in B (see also [3, page 6]). Let v be an articulation point of a connected graph G with $d_{\mathbb{BCT}(G)}(v) = 2$ whose removal disconnects G into two connected components G_1, G_2. The operation *split G at (an articulation point) v* outputs the connected graphs $G_1 \cup \{v\}, G_2 \cup \{v\}$. A 2-*tree* is a tree in which all internal vertices are of degree two.

3 Biconnected Graphs of Maximum Degree $\Delta_G \leq 2\Delta_T - 2$

Our algorithm is recursive. On a very high level, in each recursive invocation the algorithm deals with a connected graph G such that $\mathbb{BCT}(G)$ is a 2-tree. If the graph is not biconnected, then every articulation point is of degree two in $\mathbb{BCT}(G)$ and we split the graph into its biconnected components (Lemmas 5 and 6, Corollary 7). Otherwise G is biconnected and we modify G by removing edges (Lemmas 8 and 9) In either case we obtain smaller graphs (with less edges) which are analyzed recursively. A key point is to keep the graphs in a proper shape that can be maintained by the recursive calls and can be used to obtain spanning trees. Therefore we introduce *ws (well-structured) graphs* and design an algorithm that finds a low degree spanning tree in ws-graphs. Finally we apply our construction to any biconnected graph (Theorem 11).

Definition 1. A connected bipartite graph $G = (V_1, V_2, E)$ with labeling function $V_2 \to \{\boxed{0}, \boxed{1}, \boxed{2}\}$ is *ws (well-structured)* if (i) $d_G(v_1) \in \{1, 2\}$ for all $v_1 \in V_1$ and $d_G(v_2) \geq 1$ for all $v_2 \in V_2$, (ii) vertices with label $\boxed{0}$ are of even degree, those with label $\boxed{1}$ of odd degree, (iii) if there is a vertex with label $\boxed{0}$, then it is the only $\boxed{0}$-vertex and there is at most one other vertex with label $\boxed{1}$, and (iv) if there is no vertex with label $\boxed{0}$, then there are at most three vertices with label $\boxed{1}$.

Definition 2. A *ws-spanning tree* T of a ws-graph G is a spanning tree of G such that every vertex v with label $\boxed{i}$ is of degree at most $\lceil \frac{d_G(v)+i}{2} \rceil$ in T. All unlabeled vertices may be of arbitrary degree in T.

Definition 3. A *Δ_G-ws-graph* is a biconnected ws-graph with every vertex of degree at most Δ_G. A *Δ_T-ws-spanning tree* is a ws-spanning tree of a $(2\Delta_T - 2)$-ws-graph.

Remark 4. Every vertex in a Δ_T-ws-spanning tree is of degree at most Δ_T.

We define a total order $\boxed{0} \prec \boxed{1} \prec \boxed{2}$ on the labels.

Lemma 5. *Let G be a ws-graph. Let $a \in V_2$ be an articulation point with $d_{\mathbb{BCT}(G)}(a) = 2$ and with label in $\{\boxed{1}, \boxed{2}\}$. Let G_1 and G_2 be the connected graphs obtained by splitting G at a. Any of the following cases assigns labels to a in G_1 and G_2 so that G_1, G_2 are ws-graphs and the sum of any ws-spanning trees T_1 and T_2 of G_1 and G_2, respectively, is a ws-spanning tree of G.*

- *If the label of a is $\boxed{2}$ and if $d_{G_1}(a)$ and $d_{G_2}(a)$ are even,*
  then label a in one arbitrary *graph (say G_1) with $\boxed{2}$ and in the other (G_2) with $\boxed{0}$.*

- *If the label of a is $\boxed{2}$ and if $d_{G_1}(a)$ and $d_{G_2}(a)$ are odd, then label a in both graphs with $\boxed{1}$.*
- *If the label of a is $\boxed{2}$ and if $d_{G_1}(a)$ is even and $d_{G_2}(a)$ is odd, then label a in G_1 with $\boxed{2}$ and in G_2 with $\boxed{1}$.*
- *If the label of a is $\boxed{2}$ and if $d_{G_1}(a)$ is even and $d_{G_2}(a)$ is odd, then label a in G_1 with $\boxed{0}$ and in G_2 with $\boxed{2}$.*
- *If the label of a is $\boxed{1}$, then (w.l.o.g.) $d_{G_1}(a)$ is even and $d_{G_2}(a)$ is odd; label a in G_1 with $\boxed{0}$ and in G_2 with $\boxed{1}$.* $\square$

Lemma 6. *Let G be a ws-graph and let $\mathbb{BCT}(G)$ be a 2-tree. If G is not biconnected, then let B_1 and B_2 be the (only) two blocks in G that are incident to only one articulation point. Let s, t, u be a vertex with the smallest, second smallest, and third smallest label in V_2, respectively. Assume that s is not a $\boxed{2}$-vertex, that s is in B_1, and that s is not an articulation point of G. If s is a $\boxed{0}$-vertex and t is a $\boxed{1}$-vertex, then let $x = t$ be in B_2. If s, t, u are all $\boxed{1}$-vertices, then let t or u, say $x = t$, be in B_2. Otherwise, let x be an arbitrary vertex in B_2. Assume further, that x is not an articulation point in G.*

Then we can split G at all articulation points and relabel the articulation points such that the components are biconnected ws-graphs. Additionally, the sum of any ws-spanning trees of the components is a ws-spanning tree of G.

Proof. The proof is by induction on the number of blocks in G.

If there is only one block, then G is a biconnected ws-graph and we are done.

Otherwise, let a be an arbitrary articulation point that separates s from x. Split G at a into G_1 and G_2 such that s is in G_1. Then $\mathbb{BCT}(G_1)$ and $\mathbb{BCT}(G_2)$ are 2-trees. We now show how to assign labels to a in G_1 and G_2 consistently with Lemma 5 such that G_1 and G_2 fulfill all the inductive requirements of the lemma.

Let $y := \{t, u\} - \{x\}$.

If s is a $\boxed{0}$-vertex in G then a has label $\boxed{2}$ in G. Label a in G_1 by at least $\boxed{1}$ and in G_2 by at least $\boxed{0}$.

If a has label $\boxed{1}$ in G then $a = y$ and s is a $\boxed{1}$-vertex in G. Label a in G_1 and G_2 by at least $\boxed{0}$.

Otherwise, a is a $\boxed{2}$-vertex in G and s is a $\boxed{1}$-vertex in G. If G_i contains y, then label a in G_i by at least $\boxed{1}$ and in the other graph G_{3-i} by at least $\boxed{0}$.

According to Lemma 5, G_1 and G_2 are ws-graphs and the sum of their ws-spanning trees is a ws-spanning tree of G. Thus the lemma follows from the inductive hypothesis. $\square$

Corollary 7. *Let G be a connected ws-graph with maximum degree $2\Delta_T - 2$ that fulfills the requirements of Lemma 6. Then we can split G into blocks such that (i) every block is a $(2\Delta_T - 2)$-ws-graph and (ii) the sum of the Δ_T-ws-spanning trees of the blocks is a Δ_T-ws-spanning tree of G.* $\square$

Lemma 8. *Let $G = (V_1, V_2, E)$ be a ws-graph, let $|V_2| > 1$ and let s be a vertex in V_2 with $d_G(s) > 2$. Let x_1 and x_2 be two vertices adjacent to s. Let v_1 and v_2 be the other vertices than s incident to x_1 and x_2, respectively. If the graph obtained by deleting x_1 and x_2 with the incident edges and inserting the new vertex z with the edges (z, v_1) and*

(z, v_2) (all labels remain unchanged) has a ws-spanning tree T^*, then we can construct a ws-spanning tree T of G out of T^*.

Proof. Add the edges $(s, x_1), (s, x_2)$ to T^*. Additionally, if $(v_i, z) \in T^*$, $i = 1, 2$, then add (x_i, v_i) to T^*. Delete z with all incident edges. The resulting graph has exactly one cycle on which a and s lie. Thus one of the two new edges $(s, x_1), (s, x_2)$, say (s, x_1), lies on the cycle. Removal of (s, x_1) keeps the degree constraint at s and yields a ws-spanning tree T of G. □

We will call the above operation DELETE-AND-COMBINE x_1, x_2 for s and the obtained graph is denoted by $G\langle x_1, x_2, s\rangle$.

We can prove the following lemma:

Lemma 9. *Let G be a $(2\Delta_T - 2)$-ws-graph and s be the vertex with the smallest label in G. Let z be a new vertex. Let $d_G(s) \geq 4$ and let $(s, x_1), (s, x_2), (s, x_3)$ be three edges incident to s. Assume that $G - \{(s, x_1), (s, x_2), (s, x_3)\}$ has exactly one articulation point a of degree 4 in the block-cutvertex tree (Note: all other articulation points are of degree 2).*

If we split $G - \{(s, x_1), (s, x_2)\} \cup \{(z, x_1), (z, x_2)\}$ at a into two $(2\Delta_T - 2)$-ws-graphs G_1, G_2 such that s is in G_1 and z is in G_2, then we can assign labels to s and a in G_1 and to z and a in G_2 so that a Δ_T-ws-spanning tree T of G can be constructed from any Δ_T-ws-spanning trees of G_1 and G_2. □

We present a recursive algorithm that finds a Δ_T-ws-spanning tree in $(2\Delta_T - 2)$-ws-graphs. A high level description is as follows (we assume $|V_2| > 1$):

```
Algorithm $\Delta_T$-WS-SPANNING-TREE
  Input: $(2\Delta_T - 2)$-ws-graph $G$
  Output: $\Delta_T$-ws-spanning tree $T$
(1) Let $s$ be a vertex with the smallest label in $G$. If $s$ is a $\boxed{2}$-vertex and $d_G(s)$ is
    even, then label it $\boxed{0}$. Otherwise label $s$ with $\boxed{1}$.
(2) If $|V_2| = 2$ then (a); return($T$);
(3) If $d_G(s) = 2$ then (b); return($T$);
(4) Let $t$ be a vertex with the second smallest label and $u$ with the third smallest
    label. Let $(s, x_i), i = 1, 2, 3$, be three edges incident to $s$ and $(x_i, v_i)$ be the
    other edge than $(s, x_i)$ incident to $x_i$. Define $\mathcal{K} := \{v_1, v_2, v_3\}$ and $X :=$
    $\{x_1, x_2, x_3\}$.
(5) If $d_G(s) = 3$ then (c); return($T$);
(6) If $|\mathcal{K}| = 1$ then (d); return($T$);
(7) If $|\mathcal{K}| = 2$ then (e); return($T$);
(8) (f); return($T$);
```

Our algorithm considers the following cases (we present a high level description):

(a) This is the base case. Let V_2 be $\{s, v\}$ and $V_1 := \{x_1, \ldots, x_d\}$, where $d := d_G(s)$. If d is even, then the edges (s, x_1);...;$(s, x_{d/2})$;$(v, x_{d/2})$;...;(v, x_d) induce a Δ_T-ws-spanning tree T of G, because v has to be a $\boxed{2}$-vertex. If d is odd, then the edges (s, x_1);...;$(s, x_{(d+1)/2})$;$(v, x_{(d+1)/2})$;...;(v, x_d) induce a Δ_T-ws-spanning tree T of G, because v is at least a $\boxed{1}$-vertex.

(b) If $d_G(s) = 2$ then let x_1 and x_2 be the two neighbors of s. Let $v_i, i = 1, 2$, be the other neighbor of x_i than s ($v_1 \neq v_2$, because $|V_2| \neq 2$ and G is biconnected). Remove x_1, x_2 and s with the incident edges from G. If one of v_1 or v_2 is a $\boxed{1}$-vertex, say v_1, then relabel v_1 by $\boxed{0}$. Otherwise, relabel v_1 and/or v_2 with $\boxed{1}$ if they are now of odd degree. Observe that we removed the $\boxed{0}$-vertex s and that there was at most one $\boxed{1}$-vertex in G. Thus the obtained graph can be split according to Corollary 7. Let T^* be the sum of the Δ_T-ws-spanning trees of the components. Then $T := T^* \cup \{(s, x_1), (x_1, v_1), (x_2, v_2)\}$ is a Δ_T-ws-spanning tree of G.

(c) W.l.o.g. let $v_1 \neq v_2$.

If $|\{v_1, v_2, v_3\}| = 2$, then $G\langle x_1, x_2, s\rangle$ has two trivial blocks (s, x_3) and (x_3, v_3) and one non-trivial block. Thus $G\langle x_1, x_2, s\rangle$ fullfills the requirements of Corollary 7.

If $|\{v_1, v_2, v_3\}| = 3$, then we can find a vertex x_i, say x_3, such that all $\boxed{1}$-vertices are in the same block in $G - (s, x_3)$. Because t, u are in one leaf-block and v_3 in the other, the requirements of Corollary 7 are satisfied for $G\langle x_1, x_2, s\rangle$.

The construction of the Δ_T-ws-spanning tree T of G out of a Δ_T-ws-spanning tree of $G\langle x_1, x_2, s\rangle$ is done as indicated by Lemmas 6 and 8.

(d) Let $\mathcal{K}$ be $\{v\}$. Delete the vertices x_1 and x_2 together with their incident edges. The resulting graph G^* is biconnected and thus it is a $(2\Delta_T - 2)$-ws-graph. Any Δ_T-ws-spanning tree T^* of G^* can be extended to a Δ_T-ws-spanning tree T of G by adding the edges (s, x_1) and (x_2, v).

(e) Let $\mathcal{K}$ be $\{v_1, v_3\}$ and both v_1 and s be adjacent to x_1 and x_2. DELETE-AND-COMBINE x_2, x_3 for s. The resulting graph is biconnected and hence it is a $(2\Delta_T - 2)$-ws-graph. By Lemma 8, we can construct a Δ_T-ws-spanning tree T of G.

(f) In this case either the graph $G\langle x_1, x_2, s\rangle$ and/or $G\langle x_2, x_3, s\rangle$ is biconnected or $G - \{(s, x_1), (s, x_2), (s, x_3)\}$ has exactly one articulation point a of degree 4 in the block-cutvertex tree. In the first case we use Lemma 8 to get a Δ_T-ws-spanning tree T of G. In the second case we split the separating pair $\{s, a\}$ according to Lemma 9 and get a Δ_T-ws-spanning tree T of G.

Lemma 10. *Every $(2\Delta_T - 2)$-ws-graph G has a Δ_T-spanning tree. One can find a Δ_T-spanning tree of G in time $O(n^{3/2})$.*

Sketch of the proof: One can verify that if the input graph is $(2\Delta_T - 2)$-ws, then the algorithm described above always returns a Δ_T-spanning tree. Thus we only must show that the algorithm can be implemented within the required time. For the proof notice that $n \leq m \leq 2n$. We use three data structures that are dynamically maintained during the run of the algorithm.

The first data structure contains a (real) *representation of the graph*, in which each vertex keeps its incidency list. It is trivial to maintain the representation of the graph under edge deletions and insertions in constant time. Another operation which must be maintained is splitting the graph G at an articulation point a that separates vertex s from vertex x. Let G_1 and G_2 be the resulting graphs. We run two depth first search (DFS) algorithms in parallel in G, one starting from s and another from x. In each of the DFS-algorithms, we assume that the recursion stops every time we are at a. We end when one of the algorithms visits all the edges. This means that all the edges of G_1 or G_2, say G_1, have been visited. Then we create the representation of G_1 and delete the edges of G_1 from the original representation of G (to create the representation of G_2). The running time is $O(m')$, where m' is the number of edges in G_1. Since G_1 has not more edges than G_2, standard amortization argument can be used to show that the overall cost of maintaining the representation of the graph under splitting is $O(n \log n)$.

The second data structure, which is the fully dynamic data structure for maintaining biconnectivity of Rauch [21], is used for *biconnectivity queries*. It enables to perform edge deletion and insertion in amortized time $O(\sqrt{n})$, and answers in constant time the query for an articulation point that separates two specified vertices. The crucial idea is that we do not perform the splitting operation on this representation.

The third data structure is needed for *connectivity queries in the forest*. For this we use the dynamic tree data structure of Sleator and Tarjan [23].

We omit the details, but using these three data structures, one can implement Algorithm Δ_T-WS-SPANNING-TREE to run in $O(n^{3/2})$ time. Actually, the running time is $O(n \log n)$ plus the time for maintaining $O(n)$ biconnectivity updates and queries. □

Theorem 11. *Every biconnected graph $G = (V, E)$ with maximum degree $2\Delta_T - 2$ has a Δ_T-spanning tree. Such a Δ_T-spanning tree can be found in time $O(n^{3/2} + m)$.*

Proof. We first sparsify G. In [11] an $O(n + m)$-time algorithm is given that outputs a spanning subgraph G' of G that contains the same k-connected components as G and has fewer than kn edges. We use this algorithm for $k = 2$.

Now, for each edge $e = (x, y) \in E$, place a new vertex v_e in the middle of e, i.e., remove e and then add two edges (x, v_e) and (v_e, y). Let $G^* = (V^*, E^*)$ be the resulting graph. Since G' has $O(n)$ edges, G^* has $O(n)$ edges too. If we set $V_1 = \{v_e : e \in E\}$, $V_2 = V$, and assign label $\boxed{2}$ to every vertex in V_2, then G^* is a $(2\Delta_T - 2)$-ws-graph. Therefore we can find a Δ_T-spanning tree T^* of G^* by Lemma 10. We construct a Δ_T-spanning tree T of G from T^* by adding an edge $e = \{x, y\}$ iff the edges $\{x, v_e\}$ and $\{v_e, y\}$ are in T^*. One can verify that T is indeed a Δ_T-spanning tree of G. Lemma 10 ensures that the whole construction can be performed in the required running time. □

We finally notice that the running time of our algorithm may be significantly improved if we would use randomization. The fully dynamic data structure for maintaining biconnectivity of King and Rauch [22] could be applied to obtain the running time of $O(m + n \log^4 n)$ with high probability.

4 k-Connected Graphs of Maximum Degree $k(\Delta_T - 2) + 2$

Let G be a k-connected graph of maximum degree $\Delta_G \leq k(\Delta_T - 2) + 2$ and let Δ_T^* be the minimal integer such that G has a Δ_T^*-spanning tree. Fürer and Raghavachari [7] gave an algorithm for finding a spanning tree of G with maximum degree at most $\Delta_T^* + 1$. They designed a polynomial-time algorithm that finds a d-spanning tree T and a set B with vertices of degree $d-1$ in T that satisfy the following property. Let S be the set of vertices of degree d in T. If F is the forest obtained from T by removing vertices of $S \cup B$, then there are no paths in G through vertices of $V - (S \cup B)$ between different trees in F. In that case T is a spanning tree of maximum degree at most $\Delta_T^* + 1$.

The following lemma extends the result from [7] and shows that the tree found by the algorithm of Fürer and Raghavachari is of degree at most Δ_T.

Lemma 12. *Let T be an arbitrary d-spanning tree of a k-connected graph G of maximum degree $\Delta_G \leq k(\Delta_T - 2) + 2$. Let S be the set of vertices of degree d in T and let B be an arbitrary subset of vertices of degree $d - 1$ in T. Let $S \cup B$ be removed from the graph, breaking the tree T into a forest F. Suppose G satisfies the condition, that there are no paths through vertices of $V - (S \cup B)$ between different trees in F. Then $d \leq \Delta_T$.*

Proof. Let T^* be the vertex-induced subtree of T consisting of a vertex r of degree d in T and all paths from r to vertices in $S \cup B$. For every edge e of T incident to a vertex in $S \cup B$ that does not belong to any path from T^*, define a *leaf component* that consists of all vertices w of T such that the unique tree path in T from w to r contains the edge e.

There are at least $|S|(d-2) + |B|(d-3) + 2$ leaf components (This bound will be reached, if every vertex in T^* of degree at least three is either in S or in B. Otherwise there are more leaf components). Because the graph is k-connected, there must be at least k paths from each leaf component to r, where one path can go over the only vertex of $S \cup B$ incident in T to this leaf component. Hence each leaf component is incident in $G - T^*$ to at least $k - 1$ edges with endpoints in $S \cup B$. Therefore there must be at least $(k-1)\,(|S|(d-2) + |B|(d-3) + 2)$ edges of $G - T$ incident to vertices in $S \cup B$.

On the other hand, a vertex from S can be incident to at most $\Delta_G - d$ edges in $G - T$ and a vertex from B can be incident to at most $\Delta_G - (d-1)$ edges in $G - T$. Hence there may be at most $|S|(\Delta_G - d) + |B|(\Delta_G - (d-1))$ edges in $G - T$ that are incident to vertices in $S \cup B$. Therefore the following inequality must hold:

$$(k-1)\,(|S|(d-2) + |B|(d-3) + 2) \leq |S|(\Delta_G - d) + |B|(\Delta_G - (d-1))$$

We have assumed that $\Delta_G \leq k(\Delta_T - 2) + 2$. One can easily verify that this inequality holds only if $d \leq \Delta_T$. $\square$

Combining this result, initial sparsification of an input graph with $O(kn)$ edges [11], and the algorithm of Fürer and Raghavachari [7] we obtain:

Theorem 13. *Every k-connected graph G of maximum degree at most $k(\Delta_T - 2) + 2$ has a Δ_T-spanning tree. One can find such a tree in time $O(n^2\ k\ \alpha(n) \log n)$.* $\square$

5 $\mathcal{NP}$-Completeness in General Graphs

Garey, Johnson, and Tarjan [9] showed that deciding whether a 3-connected 3-regular graph has a Hamiltonian cycle (or a Hamiltonian path) is $\mathcal{NP}$-complete. We extend their result and prove that verifying whether a k-connected $k(\Delta_T - 1)$-regular graph has a Δ_T-spanning tree (for $\Delta_T = 2$ also a Hamiltonian cycle) is $\mathcal{NP}$-complete for every $k \geq 3$, $\Delta_T \geq 2$. [1]

[1] We can provide a significantly simpler proof of the $\mathcal{NP}$-completeness if we would assume that $k \neq 5$. However, similarly as in the existential proof for $\Delta_T = 2$ presented in [12], the case $k = 5$ requires more complicated arguments.

In our proof we will analyze the 3-connected 3-regular graphs used in the reduction of Garey, Johnson and Tarjan [9] of 3SAT to the Hamiltonian problem in 3-connected 3-regular graphs. We first briefly characterize all separating edge-triplets that may appear in their construction. Then we will find a special perfect matching M in the graphs constructed in [9] that will be used to build a k-edge connected k-regular graph G_k^M that has a Hamiltonian cycle (respectively a Hamiltonian path) iff the graph constructed by Garey, Johnson and Tarjan has one. By replacing each vertex in G_k^M by a $K_{k,k(\varDelta_T-1)-1}$ we will get a k-connected graph that has a $\varDelta_T$-spanning tree (respectively a Hamiltonian cycle for $\varDelta_T = 2$) iff the graph constructed by Garey, Johnson and Tarjan has one. This will imply the main result.

Using a reduction from 3SAT, Garey, Johnson, and Tarjan [9] proved the $\mathcal{NP}$-completeness of verifying whether a 3-connected 3-regular graph has a Hamiltonian cycle (or a Hamiltonian path). For every 3SAT formula $\mathcal{F}$ they construct a 3-connected 3-regular graph G (abbreviated GJT-graph) such that $\mathcal{F}$ is satisfiable iff G has a Hamiltonian cycle (or a Hamiltonian path).

Each GJT-graph is a composition of four graphs: *the Tutte-graph*, *the XOR-graph*, *the 2-OR-graph*, and *the 3-OR-graph* (see Fig. 1–4). For every clause $a \vee b \vee c$, respectively variable x, there is a construction of the form indicated by Fig. 5 and Fig. 6, respectively.

The constructions for clauses (respectively for variables/literals) are connected one after another to form a line. Then the ends of the two lines are connected via a 2-OR-graph (XOR-graph) in the case of the Hamiltonian cycle (Hamiltonian path) problem; this 2-OR- or XOR-graph will be called *the connecting 2-OR-/XOR-graph*. Each literal in each clause is connected via the XOR-graph to a cycle of the literal in the corresponding variable construction, the so-called *literal-cycle*. For an example for the formula $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee w) \wedge (y \vee \bar{z} \vee \bar{w})$ see Fig. 7. For further details of the construction of Garey, Johnson, and Tarjan we refer to [9].

We can prove the following lemma that characterizes all edge sets of cardinality three in the GJT composition that separates the graph G and are not all adjacent.

Lemma 14. *Every non-trivial separating edge triplet of a GJT-graph consists of the three a-edges of one induced Tutte-subgraph.* □

We shall use a special perfect matching M in the GJT-graphs.

Lemma 15. *Let $G = (V, E)$ be a GJT-graph. There exists a perfect matching $M \subset E$ in G, such that (i) for every separating edge quadruplet $Q \subset E$ that is minimal: $|Q \cap M| < 4$ and that (ii) for every separating edge triplet (*SET*) $S \subset E$: $|S \cap M| = 1$* □

Let $G = (V, E)$ be a k-edge connected k-regular graph. For each vertex $v \in V$, let $\varGamma_G(v) = \{\varGamma^1(v), \ldots, \varGamma^k(v)\}$ be arbitrarily ordered set of the neighbors of v in G. We define a graph $H = (V_H, E_H)$ as follows. To each vertex $v \in V$ we assign a copy of $K_{k,k(\varDelta_T-1)-1}$ that is denoted by $K_{k,k(\varDelta_T-1)-1}(v)$. Let $A(v) = \{a_1(v), \ldots, a_k(v)\}$ and $B(v) = \{b_1(v), \ldots, b_{k(\varDelta_T-1)-1}(v)\}$ be the vertices in the k and the $k(\varDelta_T - 1) - 1$ elements' vertex set of $K_{k,k(\varDelta_T-1)-1}(v)$, respectively. Let $E(v)$ denote the set of the edges in $K_{k,k(\varDelta_T-1)-1}(v)$. We define $V_H = \bigcup_{v \in V}(A(v) \cup B(v))$ and $E_H =$

$\{(a_i(v), a_j(u)) : v, u \in V, \Gamma^i(v) = u, \Gamma^j(u) = v\} \cup \bigcup_{v \in V} E(v)$. One can verify that H is of maximum degree $k(\Delta_T - 1)$ and that it is k-connected (see also [18]).

Lemma 16. *G has a Hamiltonian path iff H has a Δ_T-spanning tree.* □

Theorem 17. *For every integers k and Δ_T, $k \geq 3$, $\Delta_T \geq 2$, it is $\mathcal{NP}$-complete to verify whether a k-connected graph of maximum degree $k(\Delta_T - 1)$ has a Δ_T-spanning tree.*

Proof. For a graph G, let M be the perfect matching constructed in Lemma 15. For every $k > 2$, define the multigraph G_k^M by:
• if $3 \mid k$: duplicating each edge of G $k/3$ times.
• if $3 \mid (k-1)$: duplicating each edge of M $\lceil k/3 \rceil$ times and the other edges of G $\lfloor k/3 \rfloor$ times.
• if $3 \mid (k-2)$: duplicating each edge of M $\lfloor k/3 \rfloor$ times and the other edges of G $\lceil k/3 \rceil$ times.

One can show that G_k^M is k-edge connected and k-regular. Thus the graph obtained by replacing each vertex in G_k^M by a $K_{k,k(\Delta_T-1)-1}$ is k-connected and of maximum degree $k(\Delta_T - 1)$ [18, Theorem 3]. Additionally it has a Hamiltonian path iff G_k^M has one (Lemma 16). Now, it is easy to see that for the case $\Delta_T = 2$, the obtained graph has a Hamiltonian cycle iff G_k^M has one. □

Theorems 13 and 17 yield the following characterization of triconnected graphs.

Theorem 18. *(i) Every triconnected graph of maximum degree at most $3\Delta_T - 4$ has a Δ_T-spanning tree; such a tree can be found in polynomial time. (ii) Verifying whether a triconnected graph of maximum degree $3\Delta_T - 3$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete.* □

6 Biconnected Planar Graphs

The result from Sect. 5 does not hold for $k = 2$. Actually, Lemma 10 states that every biconnected graph of maximum degree $2(\Delta_T - 1)$ has a Δ_T-spanning tree. We can show that this bound cannot be improved even for biconnected planar graphs.

Theorem 19. *(i) Every biconnected graph of maximum degree at most $2\Delta_T - 2$ has a Δ_T-spanning tree; such a tree can be found in polynomial time. (ii) Verifying whether a biconnected planar graph of maximum degree $2\Delta_T - 1$ has a Δ_T-spanning tree is $\mathcal{NP}$-complete.* □

We finally note that our algorithm from Sect. 3 can be speeded up for planar graphs. For this notice that we have designed our algorithm in Sect. 3 so that all operations can easily preserve the embedding of a plane graph G during the modifications. Apart from operations caused by the fully dynamic biconnectivity algorithm, the algorithm in Sect. 3 requires $O((n + m) \cdot \log(n + m))$ time.

We may use the decremental data structures of [17] for maintaining biconnectivity. The main feature of that data structure is that it can perform the DELETE-AND-COMBINE operation and can determine in a fast way a separating vertex between u and v, if one exists. Thus we get the following results:

Theorem 20. *If a biconnected planar graph G is of maximum degree at most $2\Delta_T - 2$, then one can find a Δ_T-spanning tree of G deterministically in $O(n \log n)$ time using space $O(n^2)$ or in $O(n \log^2 n)$ time using space $O(n)$, and with high probability in $O(n \log n)$ time using space $O(n)$.* $\square$

Acknowledgment We are grateful to Monika Rauch Henzinger for describing some details of her fully dynamic data structures for maintaining biconnectivity.

References

1. D. Barnette. Trees in polyhedral graphs. *Canadian J. Mathematics*, 18:731–736, 1966.
2. D. Bauer, S. L. Hakimi, and E. F. Schmeichel. Recognizing tough graphs is $\mathcal{NP}$-hard. *Discrete Applied Mathematics*, 28:191–195, 1990.
3. B. Bollobás. *Extremal Graph Theory*. Academic Press, London, 1978.
4. P. M. Camerini, G. Galgiati, and F. Maffioli. Complexity of spanning tree problems, I. *European Journal of Operation Research*, 5:346–352, 1980.
5. Y. Caro, I. Krasikov, and Y. Roditty. On the largest tree of a given maximum degree in a connected graph. *Journal of Graph Theory*, 15:7–13, 1991.
6. N. Chiba and T. Nishizeki. The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *Journal of Algorithms*, 10:187–211, 1989.
7. M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17:409–423, 1994. Also in ACM-SIAM SODA 1992.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-completeness.* Freeman, New York, 1979.
9. M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar Hamiltonian circuit problem is $\mathcal{NP}$-complete. *SIAM Journal on Computing*, 5(4):704–714, 1976.
10. M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.
11. H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7:583–596, 1992.
12. B. Jackson and T. D. Parsons. On r-regular r-connected non-Hamiltonian graphs. *Bulletin of Australian Mathematics Society*, 24:205–220, 1981.
13. D. S. Johnson. The $\mathcal{NP}$-completeness column: An ongoing guide. *Journal of Algorithms*, 6:434–451, 1985.
14. S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. In *Proceedings of the 27 ACM STOC*, pp. 1–10, 1995.
15. S. Khuller, B. Raghavachari, and N. Young. Low degree spanning trees of small weight. *SIAM Journal on Computing*, 25(2):355–368, 1996.
16. S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, 1994.
17. T. Lukovski and W.-B. Strothmann. Decremental biconnectivity on planar graphs. Manuscript, 1997.
18. G. H. J. Meredith. Regular n-valent n-connected nonHamiltonian non-n-edge-colorable graphs. *Journal of Combinatorial Theory Series B*, 14:55–60, 1973.
19. V. Neumann-Lara and E. Rivera-Campo. Spanning trees with bounded degrees. *Combinatorica*, 11(1):55–61, 1991.
20. C. H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982.

21. M. Rauch. Improved data structures for fully dynamic biconnectivity. Full version. A preliminary version appeared in *Proceedings of the 26th ACM STOC*, 1994.
22. M. Rauch Henzinger and V. King. Fully dynamic biconnectivity and transitive closure. In *Proceedings of the 36th IEEE FOCS*, pp. 664–673, 1995.
23. D.D. Sleator and R.E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences* 26:362–391, 1983.
24. W.-B. Strothmann. Constructing 3-trees in 3-connected planar graphs in linear time. Manuscript, 1996.
25. S. Win. On a connection between the existence of k-trees and the toughness of a graph. *Graphs and Combinatorics*, 5:201–205, 1989.

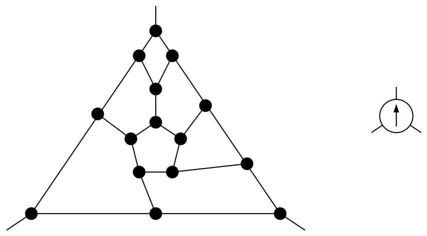
Fig. 1. Tutte-graph and its abbreviation

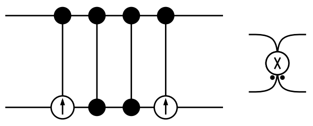
Fig. 2. XOR-graph and its abbreviation

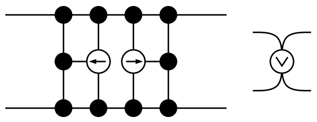
Fig. 3. 2-OR-graph and its abbreviation

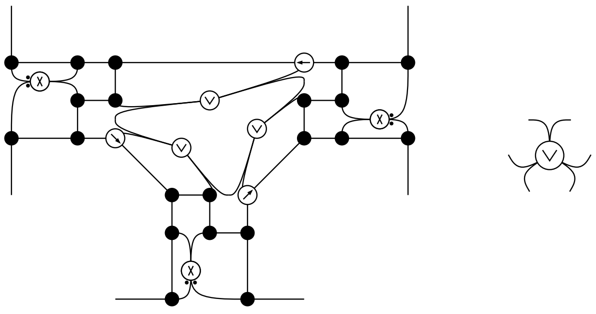
Fig. 4. 3-OR-graph and its abbreviation

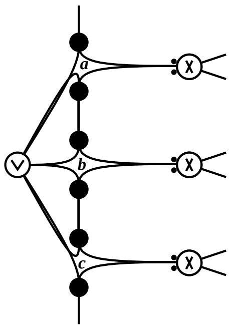

Fig. 5. Construction for a clause $a \vee b \vee c$

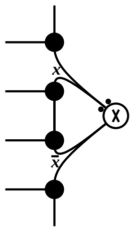

Fig. 6. Construction for a variable x

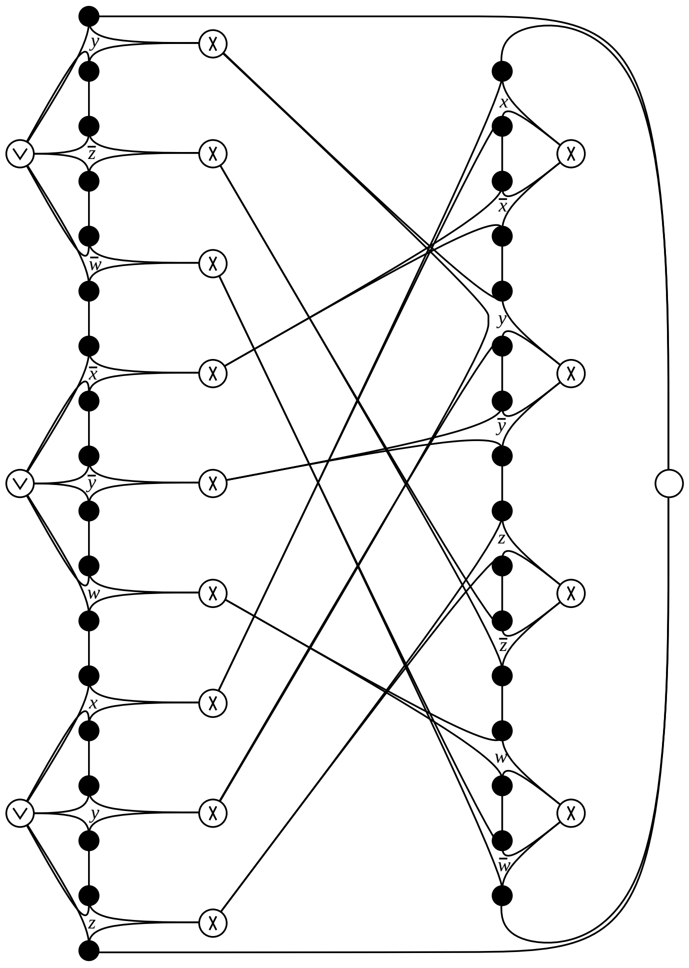

Fig. 7. GJT-graph for the formula $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee w) \wedge (y \vee \bar{z} \vee \bar{w})$