Spanning Trees with Many Leaves in Graphs without Diamonds and Blossoms

Paul Bonsma* and Florian Zickfeld**

Technische Universität Berlin, Fak. II, Str. des 17. Juni 136, 10623 Berlin, Germany
bonsma,zickfeld@math.tu-berlin.de

Abstract. It is known that graphs on n vertices with minimum degree at least 3 have spanning trees with at least $n/4+2$ leaves and that this can be improved to $(n+4)/3$ for cubic graphs without the diamond $K_4 - e$ as a subgraph. We generalize the second result by proving that every graph with minimum degree at least 3, without diamonds and certain subgraphs called blossoms, has a spanning tree with at least $(n+4)/3$ leaves. We show that it is necessary to exclude blossoms in order to obtain a bound of the form $n/3 + c$.

We use the new bound to obtain a simple FPT algorithm, which decides in $O(m)+O^*(6.75^k)$ time whether a graph of size m has a spanning tree with at least k leaves. This improves the best known time complexity for Max-Leaves Spanning Tree.

1 Introduction

This paper is concerned with finding spanning trees with maximum number of leaves. This is a well-studied problem with many applications. Different types of algorithms have been proposed for this $\mathcal{NP}$-hard problem. Constant factor approximation algorithms appear e.g. in [4,9]. In addition constructive methods exist that guarantee a certain fraction of the vertices to become leaves, when the graph satisfies certain properties [2,7,8]. Hence these results give lower bounds, which are *extremal* in the sense that examples are given which show that they are tight for their graph classes. Thirdly, FPT algorithms for the related decision problem have also received much attention, see [1,2,3,5]. These results are closely related; lower bounds have been used to find good approximations [4] and fast FPT algorithms [2,3]. Our contribution is a new extremal lower bound that generalizes and strengthens previous results, and a fast and simple FPT algorithm based on this bound.

We first introduce the extremal setting and explain our result. We then explain how this helps to obtain an improved FPT algorithm. Throughout this paper G is assumed to be a simple and connected graph on $n \geq 2$ vertices. Other graphs may be multi-graphs, disconnected, or a K_1. The minimum vertex degree of G is denoted by $\delta(G)$.

* Supported by the Graduate School "Methods for Discrete Structures" in Berlin, DFG grant GRK 1408.

** Supported by the Studienstiftung des deutschen Volkes.

E.S. Laber et al. (Eds.): LATIN 2008, LNCS 4957, pp. 531–543, 2008.

Linial and Sturtevant first observed that every graph G with $\delta(G) \geq 3$ has a spanning tree with at least $n/4+2$ leaves and that this bound is best possible (unpublished). Kleitman and West give a proof in [8] and also give stronger bounds for graphs of higher minimum degree.

The examples showing that the bound $n/4+2$ is best possible for graphs of minimum degree 3 are all obtained from a cycle by replacing every vertex by a cubic diamond. A *diamond* is the graph K_4 minus one edge, and a diamond subgraph of a graph G is a *cubic diamond* if its four vertices all have degree 3 in G, see Figure 1 (a).

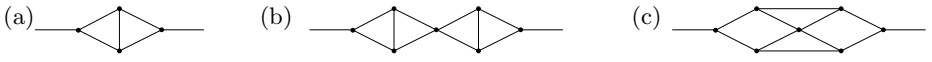

Fig. 1. A cubic diamond (a), a 2-necklace (b), and a 2-blossom (c)

Since these examples are very restricted it is natural to ask if better bounds can be obtained for graphs without cubic diamonds. This question was answered by Griggs, Kleitman and Shastri [7] for cubic graphs. They show that a cubic graph G without diamonds always admits a spanning tree with at least $n/3+4/3$ leaves and that this is best possible. For minimum degree 3 the following bound is proved in [2]. A graph G with $\delta(G) \geq 3$, without cubic diamonds, contains a spanning tree with at least $2n/7+12/7$ leaves. Replacing every vertex of a cycle by the graph from Figure 1 (b) shows that the factor $2/7$ cannot be improved. The following conjecture from [2] therefore seems natural: it was conjectured that every graph G with $\delta(G) \geq 3$ and without *2-necklaces* contains a spanning tree with at least $n/3+4/3$ leaves. Informally speaking, a 2-necklace is a concatenation of $k \geq 1$ diamonds with only two outgoing edges, see Figure 1 (b). In Definition 1 in the next section a precise definition is given. This statement, if true, would improve the bound $2n/7+12/7$ with only a minor extra restriction, and generalize the $n/3+4/3$ bound for cubic graphs from [7].

In Section 2 we disprove this conjecture by constructing graphs with $\delta(G)=3$ without 2-necklaces, which do not admit spanning trees with more than $4n/13+24/13$ leaves. On the positive side, we prove that the statement is true after only excluding one more very specific structure, called a *2-blossom*, see Figure 1 (c). A precise definition is given in Definition 2. We generalize the statement further by removing any restriction on the minimum degree. This yields Theorem 1, our main theorem, which is proved in Section 3. Let $V_{\geq 3}(G)$ denote the set of vertices in G with degree at least 3 and $n_{\geq 3}(G)$ its cardinality. Let $\ell(T)$ be the number of leaves of a graph T.

Theorem 1. *Let G be a simple, connected, non-trivial graph which contains neither 2-necklaces nor 2-blossoms. Then, G has a spanning tree T with $\ell(T) \geq n_{\geq 3}(G)/3+\alpha$, where $\alpha=2$ if $\delta(G) \leq 2$, and $\alpha=4/3$ otherwise.*

The proof is constructive and can be turned into a polynomial time algorithm for the construction of a spanning tree. The main technical contribution of this paper is that we prove this generalization of the statement in [7], and improvement of the statement in [2], without a proof as lengthy as the proofs in these two papers. This is made possible by extending the techniques and proofs from [7]. In Section 3 we argue that the long case study in [7] actually proves a strong new lemma, which we use as an important step in the proof of Theorem 1. We share the opinion expressed in [7] that a significantly shorter proof of the bound for cubic graphs might not exist. Therefore using that result in order to prove the more general statement seems appropriate.

In Section 4 we explain the consequences of Theorem 1 for FPT algorithms (short for *fixed parameter tractable*) for the following decision problem.

Max-Leaves Spanning Tree (MaxLeaf):
INSTANCE: A graph G and integer k.
QUESTION: Does G have a spanning tree T with $\ell(T) \geq k$?

When choosing k as a parameter, an algorithm for MaxLeaf is called an *FPT algorithm* if its complexity is bounded by $f(k)g(n)$, where $g(n)$ is a polynomial. See [6] for an introduction to FPT algorithms. $f(k)$ is called the *parameter function* of the algorithm. Usually, $g(n)$ will turn out to be a low degree polynomial, thus to assess the speed of the algorithm it is mainly important to consider the growth rate of $f(k)$. Bodlaender [1] constructed the first FPT algorithm for MaxLeaf with a parameter function of roughly $(17k^4)!$. Since then, considerable effort has been put in finding faster FPT algorithms for this problem. The fastest algorithms can be found in [3,5,2], which also give an overview of older results. These papers also establish a strong connection between extremal graph-theoretic results and fast FPT algorithms. The $n/4+2$ bound mentioned above is an essential ingredient in [3]. With the same techniques, the $2n/7+12/7$ bound is used in [2] to obtain the so far fastest algorithm with a parameter function in $O^*(\binom{3.5k}{k}) \subset O^*(8.12^k)$. Here the O^* notation ignores polynomial factors. Similarly Theorem 1 yields a new FPT algorithm for MaxLeaf.

Theorem 2. *There exists an FPT algorithm for* MaxLeaf *with time complexity* $O(m) + O^*(6.75^k)$*, where* m *denotes the size of the input graph and* k *the desired number of leaves.*

This algorithm is the new fastest FPT algorithm for MaxLeaf, both optimizing the dependency on the input size and the parameter function. It simplifies the ideas introduced by Bonsma, Brueggemann and Woeginger [3] and therefore is also significantly simpler than the other recent fast FPT algorithms. Hardly any preprocessing of the input graph is needed, since Theorem 1 is already formulated for a very broad graph class.

We end in Section 5 with a discussion of further improvements and applications of our results. Due to space constraints we cannot include complete proofs of Theorems 1 and 2 in this extended abstract. Instead we refer the reader to the full version of this paper for more details.

2 Obstructions for Spanning Trees with Many Leaves

As mentioned in the introduction, 2-necklaces have been identified as an obstruction for the existence of spanning trees with $n/3 + c$ leaves in graphs with minimum degree 3, see [7,8,2]. In this section we show that they are not the only such obstruction. We start by precisely defining 2-necklaces and 2-blossoms.

The degree of a vertex v in a graph G is denoted by $d_G(v)$ and by $d(v)$ if ambiguities can be excluded. A vertex v of a subgraph H of G with $d_H(v) < d_G(v)$ is called a *terminal* of H.

Definition 1 (2-Necklace). *The graph K_4 minus one edge is called a* diamond *and denoted by N_1. The degree 3 vertices are the* inner vertices *of the diamond.*

For $k \geq 2$ the diamond necklace N_k *is obtained from the graph N_{k-1} and a vertex disjoint N_1 by identifying a degree 2 vertex of N_1 with a degree 2 vertex of N_{k-1}. The two unique degree 2 vertices of N_k are denoted by c_1 and c_2.*

An N_k subgraph of G is a 2-necklace *if it only has c_1 and c_2 as terminals, which both have degree 3 in G. See Figures 1 (a) and (b). If G contains an N_1 this way, this N_1 subgraph is also called a* cubic diamond *of G.*

Definition 2 (2-Blossom). *The graph B on seven vertices shown in Figure 2 (a) is the blossom graph. A blossom subgraph B of G is a* 2-blossom *if c_1 and c_2 are its only terminals, and they both have degree 3 in G, see Figure 1 (c).*

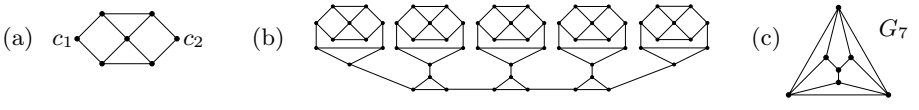

Fig. 2. A blossom, five flowers in a tree, and G_7

The two outgoing edges of 2-necklaces and 2-blossoms may in fact be the same edge, in that case G is just a 2-necklace or 2-blossom plus an edge.

The building block for the graph family that shows that the bound $n/3 + c$ cannot be achieved when only necklaces are excluded, is a graph on ten vertices called a *flower*. The graphs obtained from ternary trees by replacing inner vertices with triangles, and leaves with flowers, have no spanning tree with more than $4n/13+24/13$ leaves. Figure 2 (b) shows such a graph and it can be checked that these graphs have no more than four leaves per flower.

An important graph for our proofs, called G_7, is shown in Figure 2 (c). This is a non-cubic graph without 2-necklaces or 2-blossoms that does not admit a spanning tree with at least $n_{\geq 3}(G)/3+2$ leaves; only four leaves can be obtained. In fact, a more detailed proof of Theorem 1 shows that G_7 is the only such graph. The cubic examples from [7] show that the bound in Theorem 1 is best possible (ignoring small differences in the constant, see Section 5). These examples can also be modified to obtain extremal examples that have vertices of degree 1, 2 and 4, thus Theorem 1 is best possible in a strong sense.

3 Proof of the Main Theorem

This section is devoted to the proof of Theorem 1. The proof approach is based on [7]. We first sketch an overview of the proof of [7], then mention how we extend this method, and later give precise definitions. The two main techniques of the proof are *tree extensions* and *reduction rules*. For certain graphs G, the following approach can be used to find a tree with the desired number of leaves: start with a small tree subgraph T of G, which has the proper ratio between the number of leaves, and the number of vertices of $V_{\geq 3}(G)$ that it contains. Loosely speaking, it needs at least one leaf for every three vertices from $V_{\geq 3}(G)$. This is expressed more precisely by the value $\mathcal{P}_G(T)$ defined below, which should remain above a certain value (usually 0). Then the initial tree T is *extended* iteratively, without decreasing $\mathcal{P}_G(T)$. Proving that this is always possible is done with a case study, which considers the 'outside' of T, i.e. the subgraph of G that is not yet covered by T. When T becomes spanning, the bound follows easily from the initial lower bound on $\mathcal{P}_G$.

However there are certain substructures that are problematic for this approach. These structures are handled instead by *reduction rules*, before building the spanning tree. Every reduction rule works on a certain graph structure, and removes it. These reduction rules are applied to the graph G as long as possible, in arbitrary order. The resulting graph G' is called *irreducible*, and may be disconnected. These rules are chosen such that if a tree with the desired number of leaves is given for every component of G', it can be used to construct a tree with the right number of leaves for G. It follows that the extension procedure only has to be applied for irreducible graphs.

For our proof, it does not suffice to only consider tree extensions: it may be necessary to start new tree components, so we will actually extend a *forest* F. This is no problem when we demand an extra number of leaves for every new tree component in F. We continue with these extensions until F is a spanning forest. We add edges to obtain a spanning tree T, for which the bound will follow from $\mathcal{P}_G(F) \geq 0$.

We now state the necessary notions and lemmas more precisely, and use these to prove Theorem 1 formally. The proofs of the lemmas and details of the reduction rules are postponed to later sections, or omitted. A vertex with degree at most 2 will be called a *goober*. One important convention is that when we consider a subgraph H of G, goobers in H are always defined with respect to G, not with respect to H. In our figures, e.g. Figure 3 white vertices indicate goobers. A *high-degree vertex* is a vertex of degree at least 4.

In Section 3 the reduction rules are introduced, and it will be shown that all of them maintain the proper leaf ratio of a spanning tree, when reversed. This is expressed by the following lemma. Note that F is a maximal forest for G' if and only if it consists of a spanning tree for every component of G. A *trivial component* is an isolated vertex.

Lemma 1 (Reconstruction Lemma). *Let G' be the result of applying a reduction rule to a connected graph G, and let k be the number of non-trivial com-*

ponents of G', and $\beta \geq 0$. If G' has a maximal forest with at least $n_{\geq 3}(G')/3 + 2k - \beta$ leaves, then G has a spanning tree with at least $n_{\geq 3}(G)/3 + 2 - \beta$ leaves.

This lemma will be used with either $\beta = 0$ or $\beta = 2/3$. For showing that the graph G' in this lemma indeed has a spanning tree with at least $n_{\geq 3}(G')/3 + 2k - \beta$ leaves, it is important that the reduction rules maintain the conditions stated in Theorem 1. This is shown by the next lemma, more details about the lemma are given in Section 3. The multi-graph with two vertices and two parallel edges is denoted by $K_2 + e$.

Lemma 2. *Let G' be obtained from a simple, connected graph G without 2-necklaces or 2-blossoms by the application of a reduction rule. Then (i) G' is connected, or every component of G' contains a goober, and (ii) every component of G' is either simple or it is a $K_2 + e$, and (iii) G' contains neither 2-necklaces nor 2-blossoms, and (iv) if G contains a goober, then G' contains a goober.*

We now state the definitions and lemmas used in proving extendibility of a forest subgraph. When F and G are graphs, $F \subseteq G$ and $F \subset G$ denote the subgraph resp. proper subgraph relation. Let $F \subseteq G$. By $n_G(F)$ we denote the number of non-goober vertices of G that are in $V(F)$. By $\ell_d(F)$ we denote the number of *dead leaves* of F, that is leaves of F, which have no neighbor in $V(G) \setminus V(F)$. Let $cc(F)$ denote the number of connected components of F.

Definition 3 (Leaf-Potential). *The* leaf-potential *of a subgraph $F \subseteq G$ is* $\mathcal{P}_G(F) = 2.5\ell(F) + 0.5\ell_d(F) - n_G(F) - 6cc(F)$.

We now show that if G has a spanning subgraph F with $\mathcal{P}_G(F) \geq 0$, then a tree satisfying the desired bound exists. We may assume that F is a forest. Since all leaves of F are dead, we have $0 \leq \mathcal{P}_G(F) = 3\ell(F) - n_{\geq 3}(G) - 6cc(F)$, and thus $\ell(F) \geq n_{\geq 3}(G)/3 + 2cc(F)$. We can now add $cc(F) - 1$ edges to F to obtain a spanning tree T, losing at most $2(cc(F) - 1)$ leaves, so $\ell(T) \geq n_{\geq 3}(G)/3 + 2$.

Definition 4 (Extendible). *Let F be a subgraph of a graph G. Then F is called* extendible *if there exists an F' with $F \subset F' \subseteq G$ and $\mathcal{P}_G(F') \geq \mathcal{P}_G(F)$, and F' is called an* extension.

Above we already informally mentioned the subgraph of G 'outside' a subgraph $F \subset G$. This graph may formally be defined as an edge induced graph as follows.

Definition 5 (Graph Outside F). *Let F be a non-spanning subgraph of G. The* subgraph of G outside of F *is* $F^C = G[\{uv \in E(G) : u \notin V(F)\}]$.

Note that no edges between two vertices that are both in $V(F)$ are included in F^C. We can now formulate the two lemmas that together show that every forest subgraph in an irreducible graph without 2-necklaces and 2-blossoms is extendible. More details are given in Section 3.

Lemma 3 (Start Lemma). *Let $G \neq G_7$ be an irreducible graph, and let F be a (possibly empty) subgraph of G such that F^C contains at least one high-degree vertex and contains neither 2-necklaces nor 2-blossoms. Then F is extendible.*

Lemma 4 (Extension Lemma). *Let G be an irreducible graph, and let F be a non-empty subgraph of G such that F^C has maximum degree 3 and contains no 2-necklaces. Then F is extendible.*

The following theorem appears in [7] as Theorem 3 (reformulated slightly for our purposes).

Theorem 3. *Every simple, connected, irreducible graph G of maximum degree exactly 3 has a spanning tree with at least $n_{\geq 3}(G)/3 + \alpha$ leaves, where $\alpha = 4/3$ if G is cubic and $\alpha = 2$ otherwise.*

We now have collected all the necessary tools to prove Theorem 1.

Proof of Theorem 1. We prove, by induction over the number of edges, that every simple, connected, non-trivial graph G without 2-necklaces or 2-blossoms, has a spanning tree T with $\ell(T) \geq n_{\geq 3}(G)/3 + \alpha$, where $\alpha = 4/3$ if $\delta(G) \geq 3$, and $\alpha = 2$ if $\delta(G) \leq 2$.

First suppose G is irreducible. If G has maximum degree exactly 3, Theorem 1 follows immediately from Theorem 3. If G has maximum degree at most 2, G has a spanning tree with at least two leaves (since we assumed that G is not a K_1), which suffices.

If $G = G_7$, then a spanning tree with $4 = n_{\geq 3}(G)/3 + 5/3$ leaves can be obtained. So we may now assume that G contains at least one high degree vertex, and is not equal to G_7.

We start with an empty subgraph F of G which has $\mathcal{P}_G(F) = 0$. The Start Lemma (Lemma 3) shows that, as long as there is at least one high degree vertex not in F, we can extend F while maintaining $\mathcal{P}_G(F) \geq 0$. When all high degree vertices are included in F, the Extension Lemma (Lemma 4) can be applied iteratively, until a spanning subgraph F' is obtained with $\mathcal{P}_G(F') \geq 0$. By our observation following Definition 3, it then follows that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 2$ leaves.

It remains to consider the case that G is reducible (the induction step). We can apply a reduction rule, such that the resulting graph G' again contains no 2-necklaces or 2-blossoms, and such that every component is either simple or a $K_2 + e$ (Lemma 2). First suppose G' is connected. By Lemma 2, if $\delta(G) \leq 2$, then $\delta(G') \leq 2$, and by induction G' has a spanning tree with at least $n_{\geq 3}(G')/3 + 2$ leaves. Lemma 1 then shows that G admits a spanning tree with at least $n_{\geq 3}(G)/3 + 2$ leaves. Similarly, if $\delta(G) \geq 3$ then it follows that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 4/3$ leaves. Now suppose the reduction rule yields a disconnected graph G'. Then every resulting component has a goober (Lemma 2). So by induction, every non-trivial component C of G' has a spanning tree with at least $n_{\geq 3}(C)/3 + 2$ leaves. Thus Lemma 1 implies that G has a spanning tree with at least $n_{\geq 3}(G)/3 + 2$ leaves. □

Reducible Structures. We now present the reduction rules. We introduce five reduction rules that will be called the *high-degree reduction rules*. Each consists of an operation on a certain subgraph, and conditions for when it may be applied.

Figure 3 shows the graph operations for the five rules. For each rule, on the left the subgraph is shown that is reduced by the rule, and on the right the resulting subgraph, which has the same terminal set. The encircled vertices are the terminals, which may have further incidences, unlike the other vertices. In some cases outgoing half edges are added to indicate conditions on minimum vertex degrees. None of the vertices in the figures may coincide, but there are no restrictions on outgoing edges sharing end vertices. Goobers are shown as white vertices, but in the case of (R3), vertex v or w may also become a goober, depending on the original degree. The numbers above the arrows indicate the decrease in $n_{\geq 3}$.

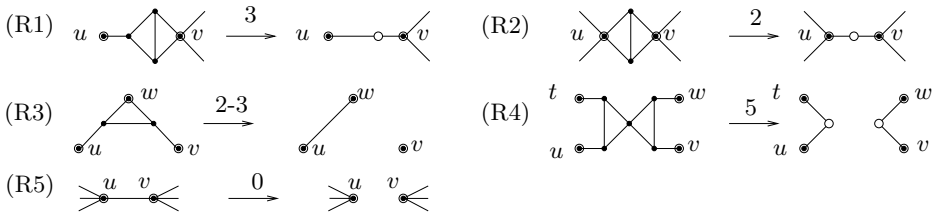

Fig. 3. The high-degree reduction rules

The following restrictions are imposed on applying these operations to a graph G. First, no reduction rule may be applied if (i) it introduces multi-edges that are incident with a non-goober, or if (ii) it introduces a 2-necklace or 2-blossom. In addition, the following rule-specific restrictions are imposed: for (R1), $d_G(v) \geq 4$ is required. For (R2), $d_G(u) \geq 4$ and $d_G(v) \geq 4$ are required. Rule (R3) may not disconnect a component, and may introduce at most one new goober. Rule (R4) may only be applied if it *does* disconnect a component. For (R5), $d_G(u) \geq 4$ and $d_G(v) \geq 4$ are required, and uv may not be a bridge. A *bridge* is an edge which upon deletion increases the number of components.

Considering these conditions on the applicability of the rules, it is obvious that Lemma 2 holds for the high-degree reduction rules. For the proof of Lemma 1, consider Figure 4. This figure shows the *tree reconstructions* for the high-degree reduction rules. If the application of a reduction rule on G gives a graph G', then without loss of generality, a spanning tree T' of G' has one of the forms shown on the left. On the right it is shown how to adapt T' to obtain a spanning tree of G. Dashed edges are present in the resulting tree if and only if they are part of T'. For the rules (R1), (R2) and (R3), one more leaf is gained, which is enough since $n_{\geq 3}(G) - n_{\geq 3}(G') \leq 3$ for these rules and G' is connected, i.e. $k = 1$. For rule (R4), no leaves are gained, but (R4) disconnects the graph and $k = 2$. Thus the increase of $n_{\geq 3}$ by 5 is compensated since each of the two components brings with it an additive term of 2. For (R5), nothing has to be proved, so Lemma 1 holds for the high-degree reduction rules.

Besides the rules (R1)-(R5) we use the seven reduction rules that are defined in [7], and we call them the *low-degree reduction rules*. These rules are shown in

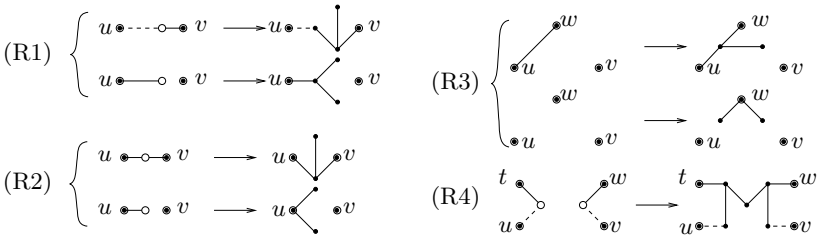

Fig. 4. Spanning tree constructions when reversing the new reduction rules

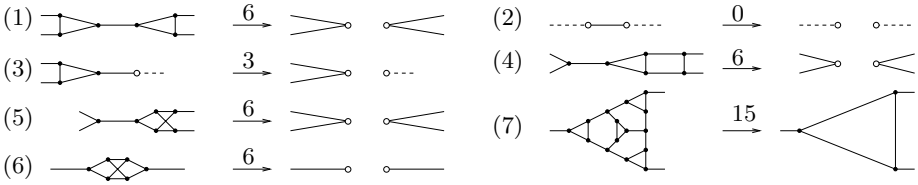

Fig. 5. The seven low-degree reduction rules

Figure 5. Conditions on their applicability are given in [7] which ensure that both Lemmas 1 and 2 hold for them. A graph to which none of the twelve reductions can be applied is called *irreducible*.

Extension Lemmas. We now sketch the proofs of Lemmas 3 and 4, which handle the construction of an extension F' of a subgraph $F \subseteq G$. Lemma 3 is proved by considering all possible neighborhoods of a high-degree vertex v of G that is not yet included in F. For every possibility, we can either give an extension of the forest that does not decrease $\mathcal{P}_G$, or we can identify a reducible structure around v, which is a contradiction with the irreducibility of G. For details of this case study we refer to the full version of this paper. We first consider the cases where v is at distance at most two of a vertex that is already in F. These cases are easily handled by extending existing trees. However when v is at a larger distance from F, we instead build a new tree around v, and add it to the forest F. Note that for instance adding a star consisting of non-goobers around a vertex of degree 5 increases $\mathcal{P}_G$ by $2.5 \cdot 5 - 6 - 6 = 0.5$, which thus gives a valid extension. Higher degrees and goobers increase $\mathcal{P}_G$ even further. For degree 4 vertices v, more cases need to be considered, but a valid extension can always be found unless v is part of two edge-disjoint triangles and all its neighbors have degree 3. However, most of these cases are reducible, provided v is not the center of a 2-blossom. In the remaining cases an extension can be found.

We now argue that the long case study in [7] in fact proves the Extension Lemma (Lemma 4). First of all we remark that in [7], goobers are defined differently, namely as vertices of degree at most two *resulting from reduction rules*. Considering the reduction rules, it can be seen that this extra condition adds no

information (for instance about the possible neighborhoods of goobers). Indeed, no such information is used in the proofs in [7], and thus goobers may simply be defined as we do. The case study in Section 4 of [7] proves the following statement, expressed using our notations.

Lemma 5. *Let G be a graph with maximum degree exactly 3, without diamonds, that is irreducible with respect to the low-degree reduction rules. Let F be a non-empty tree subgraph of G. Then there exists a tree F' with $F \subset F' \subseteq G$ and*

$$2.5(\ell(F') - \ell(F)) + 0.5(\ell_d(F') - \ell_d(F)) - (n_G(F') - n_G(F)) \geq 0.$$

The most important observation is that nowhere in the case study that proves Lemma 5, any information about the current tree F is used; only information about what we defined as F^C is used. In particular, the fact that F is connected is never used in the proof, and neither are upper bounds on degrees of vertices already included in F. So it suffices to state the maximum degree 3 condition for F^C, and the condition that F is a tree may be removed. Furthermore, an irreducible graph with maximum degree 3 that contains no 2-necklaces does not contain any diamonds as subgraphs. So we may replace the 'no diamond' condition by the 'no 2-necklace' condition. Our definition of irreducible implies irreducibility with respect to the low-degree reduction rules, so this change is also not a problem. Finally, the graph F' that is constructed by Lemma 5 has the same number of components as F, so the expression in Lemma 5 simply means that $\mathcal{P}_G(F') \geq \mathcal{P}_G(F)$. Altogether this yields Lemma 4.

4 A Fast FPT Algorithm for MaxLeaf

In this section we present a fast and relatively simple FPT algorithm for MaxLeaf, which uses Theorem 1 as an essential ingredient. The other two ingredients are a short preprocessing step, consisting of two reduction rules, and an enumerative procedure, which is similar to the one introduced in [3], and also applied in [2].

We start by presenting the two reduction rules that constitute the preprocessing phase. Recall that in 2-necklaces and 2-blossoms both terminals have degree 3 in G. The rules we introduce now also reduce diamonds and blossoms whose two terminals have arbitrary degree. However the two terminals of the subgraph must still be the two vertices that have degree 2 in the diamond necklace or blossom itself. Such a subgraph of G will be called a *2-terminal diamond* respectively a *2-terminal blossom*. Rule (F1) in Figure 6, which resembles rule (R2), reduces 2-terminal diamonds. Since a 2-necklace N_k consists of k 2-terminal diamonds, those are reduced as well by rule (F1). Rule (F2) in Figure 6 reduces 2-terminal blossoms.

Lemma 6. *Let G' be the result of applying reduction (F1) or (F2) to G. Then $(G', k-1)$ is a YES-instance for* MaxLeaf *if and only if (G, k) is a YES-instance for* MaxLeaf*.*

Throughout this section we will denote the set of leaves of a graph G by $L(G)$. We now explain how to obtain a graph $S(G)$ from a graph G by suppressing

Fig. 6. Two reduction rules for an instance (G, k) of MaxLeaf

vertices. *Suppressing* a vertex u of degree 2 means deleting u and adding an edge between the two end vertices of the incident edges. We allow this operation to introduce parallel edges and loops, so the degrees of non-suppressed vertices are maintained. If $n_{\geq 3}(G) = 0$, that is G is a path or cycle, then $S(G)$ is the empty graph. If $n_{\geq 3}(G) > 0$ then $S(G)$ is obtained from G by suppressing all degree 2 vertices. So $V(S) = L(G) \cup V_{\geq 3}(G)$, and G is a subdivision of $S(G)$. Hence loops and non-loop edges of $S(G)$ correspond to cycles and paths of G respectively. Let uv be a non-loop edge of $S(G)$ where the corresponding path P_{uv} in G has i internal vertices. We define a cost function c on the non-loop edges of $S(G)$ which assigns cost $c(uv) = \min\{i, 2\}$ to uv. Thus $c(uv)$ is the maximum possible number of leaves that a spanning tree of G can have among the internal vertices of P_{uv}. Now we are ready to present the FPT algorithm in Algorithm 1.

Algorithm 1. An FPT algorithm for MaxLeaf

INPUT: a MaxLeaf instance (G, k).

1) **while** G has a 2-terminal diamond or 2-terminal blossom subgraph **do**
   $G :=$the result of applying (F1) or (F2) to G
   $k := k - 1$
2) **if** $n_{\geq 3}(G) \geq 3k$ or $|L(G)| \geq k$ or $k \leq 2$ **then** return(YES)
3) construct $S(G)$ and c
4) **for** all $L \subseteq V_{\geq 3}(G)$ with $|L| \leq k$ **do**
   if G has a spanning tree T with $L \subseteq L(T)$ and $|L| + |L(T) \backslash V_{\geq 3}(G)| \geq k$ **then** return(YES)
5) return(NO)

The decision in Step 4 can be made in polynomial time in the size of $S(G)$. The essential step is to solve a minimum weight spanning tree problem on $S(G) - L$, using edge costs c. We omit the proof of the following lemma's, noting that the algorithm is similar to the ones in [3] and [2].

Lemma 7. *Let (G, k) be a* MaxLeaf *instance for which $S(G)$ and c are non-empty and known. For any $L \subseteq V_{\geq 3}(G)$, deciding whether G has a spanning tree T with $L \subseteq L(T)$ and $|L| + |L(T) \backslash V_{\geq 3}(G)| \geq k$ can be done in time polynomial in the size of $S(G)$.*

Lemma 8. *Algorithm 1 returns YES if and only if its input (G, k) is a YES-instance.*

Proof sketch for Theorem 2. It only remains to prove the complexity bound. The first three steps can be done in linear time by building the proper data structures. For this it is essential that the degree of non-terminal vertices of 2-terminal diamonds and blossoms is bounded by a constant. We omit the details. Since the reductions in Step 1 do not increase the number of vertices or the value of k, we may assume that n and k are the number of vertices and parameter of the reduced instance, as it is after Step 1.

Step 4 of the algorithm is only executed when $n_{\geq 3}(G) < 3k$ and $|L(G)| < k$. Furthermore $V(S(G)) = L(G) \cup V_{\geq 3}(G)$, so every iteration of the for-loop of Step 4 takes time polynomial in k (Lemma 7). This for-loop is executed once for every subset $L \subseteq V_{\geq 3}(G)$ with $|L| \leq k$. Using $|V_{\geq 3}(G)| \leq 3k$, the number of such sets can be verified to be $O(k\binom{3k}{k})$. Using Stirling's approximation $x! \approx x^x e^{-x}\sqrt{2\pi x}$, we obtain that the loop is executed $O^*(6.75^k)$ times. □

5 Conclusions

We conclude with some remarks about possible improvements and further applications. Theorem 1 can be strengthened at the cost of lengthier proofs. A full version of this paper shows that $n_{\geq 3}(G)/3 + c$ leaves can be obtained where $c = 4/3$ when $G = Q_3$, the 3-dimensional cube, $c = 5/3$ when $G = G_7$ or $G \neq Q_3$ is cubic, and $c = 2$ otherwise. In addition it can be shown that any graph G has a spanning tree with at least $(n_{\geq 3}(G) - x - y)/3 + c$ leaves, where x is the number of 2-necklaces in G and y is the number of 2-blossoms in G. This is a strong statement since firstly it holds for all graphs (barring the trivial conditions that G should be simple, connected and non-trivial), and secondly it not only generalizes the bound from [7], but also the $n/4 + 2$ bound for graphs with minimum degree three, and the $2n/7 + 12/7$ bound from [2] (see Section 1), when substituting the appropriate upper bounds for x and y. The usefulness of bounds of this form was recently demonstrated in [4], where a similar bound was used to obtain an improved approximation algorithm.

Theorem 1 can be used to show that the 'flower tree' example from Section 2 is extremal: when only 2-necklaces are forbidden, it is always possible to obtain at least $4n_{\geq 3}(G)/13 + 24/13$ leaves in non-cubic graphs.

References

1. Bodlaender, H.L.: On linear time minor tests with depth-first search. J. Algorithms 14(1), 1–23 (1993)
2. Bonsma, P.S.: Sparse cuts, matching-cuts and leafy trees in graphs. PhD thesis, University of Twente, Enschede, the Netherlands (2006), `http://purl.org/utwente/57117`
3. Bonsma, P.S., Brueggemann, T., Woeginger, G.J.: A faster FPT algorithm for finding spanning trees with many leaves. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 259–268. Springer, Heidelberg (2003)

4. Correa, J.R., Fernandes, C.G., Matamala, M., Wakabayashi, Y.: A 5/3-approximation for finding spanning trees with many leaves in cubic graphs. In: WAOA 2007 (to appear, 2007)
5. Estivill-Castro, V., Fellows, M.R., Langston, M.A., Rosamond, F.A.: FPT is P-time extremal structure I. In: ACiD 2005. Texts in algorithmics, vol. 4, pp. 1–41. King's College Publications
6. Flum, J., Grohe, M.: Parameterized complexity theory. Springer, Berlin (2006)
7. Griggs, J.R., Kleitman, D.J., Shastri, A.: Spanning trees with many leaves in cubic graphs. J. Graph Theory 13(6), 669–695 (1989)
8. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discrete Math. 4(1), 99–106 (1991)
9. Solis-Oba, R.: 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 441–452. Springer, Heidelberg (1998)