A 3/2-Approximation Algorithm for Finding Spanning Trees with Many Leaves in Cubic Graphs

Paul Bonsma* Florian Zickfeld †

Technische Universität Berlin, Institut für Mathematik, Sekr. MA 5-1,
Straße des 17. Juni 136, 10623 Berlin, Germany
bonsma,zickfeld@math.tu-berlin.de

March 28, 2008

Abstract

We consider the problem of finding a spanning tree that maximizes the number of leaves (MaxLeaf). We provide a 3/2-approximation algorithm for this problem when restricted to cubic graphs, improving on the previous 5/3-approximation for this class. To obtain this approximation we define a graph parameter $x(G)$, and construct a tree with at least $(n-x(G)+4)/3$ leaves, and prove that no tree with more than $(n-x(G)+2)/2$ leaves exists. In contrast to previous approximation algorithms for MaxLeaf, our algorithm works with *connected dominating sets* instead of constructing a tree directly. The algorithm also yields a 4/3-approximation for Minimum Connected Dominating Set in cubic graphs.

1 Introduction

The problem MaxLeaf is defined as follows: given a connected graph G, find a spanning tree of G that maximizes the number of leaves. This problem is NP-hard [11], even for cubic graphs [16]. It is closely related to the problem MinCD-Set, which asks for a smallest possible connected dominating set or *CD-set*, which is a set $S \subseteq V(G)$ such that $G[S]$ is *connected*, and every vertex of G is either in S or adjacent to S (a *dominating set*). Observing that the non-leaves of a spanning tree form a CD-set, it is easily seen that G has a spanning tree with at least k leaves if and only if G has a CD-set of size at most $|V(G)| - k$ (provided that $G \neq K_2$), and that these can be constructed from each other in polynomial time.

These problems have many theoretical and practical applications; in particular, recently they have received a lot of attention due to their importance in wireless networks [5]. Therefore it is not surprising that they have been considered from many different viewpoints, such as purely combinatorial settings (see below), and using most of the different algorithmic paradigms for solving hard problems, such as approximation algorithms (see below), fixed parameter tractable algorithms [4, 8] and fast exact algorithms [9]. Generalizations such as to directed graphs have been studied [1]. Restrictions to different graph classes have also been considered. Motivated by the wireless networking applications, unit disk graphs have been widely studied [5]. In this paper, we study the two problems when restricted to cubic graphs, which was done before in [6, 12, 16, 17]. Let $n(G)$ denote $|V(G)|$, and let $\delta(G)$ and

*Supported by the Graduate School "Methods for Discrete Structures" in Berlin, DFG grant GRK 1408
†Supported by the Studienstiftung des deutschen Volkes

$\Delta(G)$ denote the minimum and maximum degree of G, respectively. If there is no cause for confusion we will simply write n, δ and Δ. $\Delta(G)$

With regard to the approximability of MaxLeaf, it is know that a polynomial time approximation scheme is unlikely to exist, since the problem is known to be MAX SNP-complete [10]. A 3-approximation was given by Lu and Ravi [18], and later a 2-approximation was given by Solis-Oba [20], which is the current best approximation ratio for general graphs. Loryś and Zwoźniak initiated the study of approximation algorithms for MaxLeaf in cubic graphs, and gave a 7/4-approximation for this class [17]. This was recently improved to 5/3 by Correa et al [6]. In this paper, we will give a 3/2-approximation for MaxLeaf for cubic graphs. From an approximation viewpoint MaxLeaf and MinCD-Set behave quite differently: Guha and Khuller [14] showed that it is unlikely that constant factor approximation algorithms exist for MinCD-Set for general graphs. The current best approximation is $2 + \ln \Delta(G)$, given by Ruan et al [19]. For cubic graphs our algorithm will approximate MinCD-Set with a guarantee of 4/3.

In another branch of research, a number of tight lower bounds is given for the maximum number of leaves that can be obtained, for (connected) graphs from different classes. Linial and Sturtevant first proved that every graph with $\delta \geq 3$ has a spanning tree with at least $n/4 + 2$ leaves (unpublished). A short proof appears in [15], where it is also shown that in graphs with $\delta \geq 4$, $2n/5 + 8/5$ leaves can be obtained. For graphs with $\delta \geq 5$, $n/2 + 2$ leaves are possible [13]. The $n/4 + 2$ bound is also tight for cubic graphs, but when in addition *diamonds* are forbidden as subgraphs, Griggs et al [12] showed that $n/3 + 4/3$ leaves can be obtained. A *diamond* is a K_4 minus one edge. Recently it was shown that when in addition to diamonds, a certain subgraph on seven vertices is forbidden, the $n/3 + 4/3$ bound also holds for graphs with $\delta \geq 3$ [4, 21]. This generalizes an earlier result [2, 3] that proves the same bound for graphs with $\delta \geq 3$ without triangles. However, because of some useful features (see Section 2), it is this earlier result that we will apply in this paper. In [21], a number of these bounds and similar bounds have been generalized to graphs with arbitrary degrees. Even though the algorithmic viewpoint is not stressed in the results mentioned above, all proofs easily give polynomial time algorithms that construct a tree satisfying the bounds. *diamond*

This leads us back to approximation algorithms. For instance, note that the $n/4 + 2$ bound gives a trivial 4-approximation when MaxLeaf is restricted to graphs with $\delta \geq 3$. It is straightforward to show that spanning trees in cubic graphs have at most $n/2 + 1$ leaves. Combined with the $n/4+2$ lower bound, this gives a trivial 2-approximation for cubic graphs. The 5/3-approximation given by Correa et al [6] is based on a more sophisticated version of this idea which we will now treat in more detail, since we use a similar strategy.

The goal of [6] was to match the upper bound $n/2 + 1$ for cubic graphs with the lower bound $n/3+4/3$ for cubic graphs *without diamonds*. To make this work, only diamonds have to be treated in some way. This was done by defining a graph parameter $c(G)$ that depends on the number and positions of diamonds in G: for a subgraph H of G, the *internal vertices* of H are those with only neighbors in H. Note that diamonds in cubic graphs always have two internal vertices. The parameter $c(G)$ now denotes the number of components obtained when removing all internal vertices of diamonds from G. Using the bound from [12], it was shown that a spanning tree with at least $(3n - 2c(G) + 17)/10 > \frac{3}{10}(n - 2c(G) + 4)$ leaves can be constructed, and it was shown that any spanning tree has at most $(n - 2c(G) + 4)/2$ leaves. Together this gives a $\frac{1}{2}/\frac{3}{10} = \frac{5}{3}$ approximation. It was also conjectured in [6] that a 3/2-approximation algorithm is possible for MaxLeaf. *internal vertices*

Our contribution We prove this conjecture by providing a 3/2-approximation for MaxLeaf in cubic graphs. The algorithm itself is very simple, although the analysis is more involved. It is necessary to extend the study of problematic structures further, beyond only diamonds. This is indicated by the examples from [6]: there it is shown that graphs G with $c(G) = 0$ exist that have no spanning tree with more than $\lceil(3n + 17)/10\rceil$ leaves, hence considering only the parameter $c(G)$ is not good enough. We consider all triangles of G, identify different types of them, and use their positions in G to define a number of graph parameters. This is done in Section 3. Then in Section 4, we state the algorithm, and prove it yields at least $(n - x(G) + 4)/3$ leaves, where $x(G)$ is a combination of the defined graph parameters. In Section 5 we prove that spanning trees in cubic graphs have at most $(n - x(G) + 2)/2$ leaves. Together, this yields the 3/2-approximation for MaxLeaf, and the 4/3-approximation for MinCD-Set (see Section 6). We believe that the two bounds we prove, and the identified graph parameters are interesting in their own right, showing exactly in which cases diamonds and triangles make it hard or even impossible to find spanning trees with at least $n/3 + 4/3$ leaves in graphs with $\delta \geq 3$.

Unlike the previous algorithms, our algorithm is in fact an algorithm that constructs a CD-set instead of a tree. In [2, 3] it was proved that in graphs with $\delta \geq 3$ without triangles, any CD-set S that satisfies some simple properties has $|S| \leq 2n/3-4/3$. In Section 7 we show that this bound can be generalized to graphs with triangles, but with a correction term that (sloppily speaking) depends on the number of triangles in $G[S]$. Our algorithm constructs S such that it contains at most $x(G)$ triangles, such that the bound yields $|S| \leq (2n+x(G)-4)/3$. This in turn gives the bound for spanning trees mentioned above. More details on the bounds for CD-sets are given in Section 2, together with some basic definitions.

2 Preliminaries

Basic notations and terminology For basic graph theoretic notions we refer to [7]. We assume all graphs to be simple, with the exception that we allow edge contractions to yield parallel edges and loops. When applying edge contractions, edges are assumed to be labeled, that is, edge identities are preserved even if the labels of their end vertices change.

The set resulting from removing element v from S is denoted by $S - v$, and adding an element is denoted by $S + v$. For a set $S \subseteq V(G)$, we use $N(S)$ to denote the set of all vertices that have a neighbor in S, and $\overline{S} = V(G)\backslash S$. The number of components of a graph G is denoted by $cc(G)$. *Internal vertices* of a subgraph H of G are those vertices that only have neighbors in H. Let $\text{Int}(H)$ denote the set of internal vertices of H (with respect to its supergraph G). We say that a subgraph H of G is a *block* of G if it is a maximal 2-connected subgraph. Note that in contrast to the usual definition, this implies that bridges of G are not part of any block. The vertex degree of $v \in V(G)$ is denoted by $d_G(v)$, or $d(v)$ if possible.

Minimal CD-sets Firstly it is important to recall that a set is said to be *minimal* for some set of properties whenever it has no strict subset with these properties. Sets of minimum cardinality that satisfy a set of properties are called *minimum*. A CD-set S is called a *2-CD-set* if every vertex in $\overline{S}$ has at most two neighbors in S. *2-CD-set*

Proposition 1 *In cubic graphs, for any 2-CD-set S, a minimal 2-CD-set $S^* \subseteq S$ can easily be found in polynomial time, by iteratively removing single vertices or pairs of vertices while maintaining a 2-CD-set.*

Formulated just for cubic graphs, the results in [2, 3] yield the following bound.

Theorem 2 *Let $G = (V, E)$ be a connected cubic graph. Let S be a minimal 2-CD-set of G where $G[S]$ contains no triangles, and let $S' \subseteq S$ be a minimal CD-set of G. Then $|S'| \leq (2n(G) - 4)/3$.*

This theorem can be applied to graphs without triangles, in that case it holds for any minimal 2-CD-set. But when considering the proof in [2, 3], it can be seen that actually the following stronger statement is proved.

Theorem 3 *Let $G = (V, E)$ be a connected cubic graph. Let S be a minimal 2-CD-set of G where $G[S]$ has b^{Δ} blocks that contain triangles, and let $S' \subseteq S$ be a minimal CD-set of G. Then $|S'| \leq (2n(G) + b^{\Delta}(S') - 4)/3$.*

For the convenience of the reader, we have included a new concise proof of Theorem 3 in Section 7. This is shorter than the one in [2, 3], partly because it is only formulated for cubic graphs, and partly because of a different setup of the proof.

The strength of Theorem 3 lies not in the combination of the bound and the graph class itself (as we remarked in the introduction, in that sense it is strengthened and generalized for instance by the bound from [4]), but in the fact that it holds for *any minimal 2-CD-set*. This allows us to first construct a 2-CD-set S^* that satisfies some useful properties, namely that it contains few triangles, and then consider a minimal 2-CD-set $S \subseteq S^*$.

3 Subgraphs obtained by removing triangles

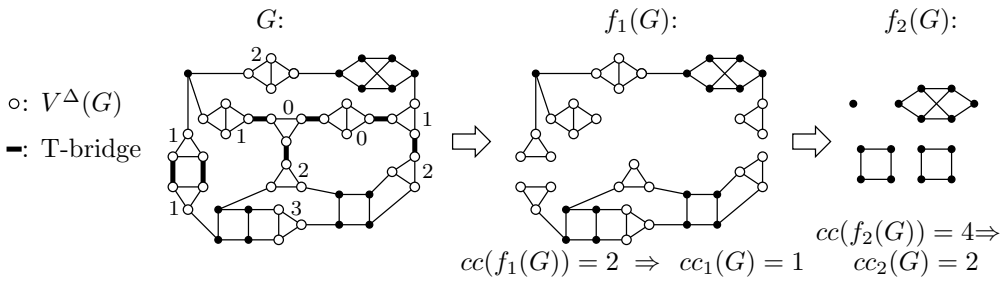

Figure 1: An example of G, $f_1(G)$ and $f_2(G)$.

The operations and notions defined in this section are illustrated in Figure 1. This figure introduces the example of G that we will use to illustrate most proofs in the paper. For a graph G with $\Delta(G) \leq 3$, let $V^{\Delta}(G) \subseteq V(G)$ be those vertices of G that are part of a triangle. $V^{\Delta}(G)$

We distinguish a number of triangle types of G. First we distinguish between triangles of G that are part of diamonds, and those that are not. **From now on, when we talk about *triangles of* G, we mean those triangles that are not part of diamonds**, except when explicitly noted otherwise. Note that since $\Delta(G) \leq 3$, all diamonds of G are pairwise vertex disjoint, and all triangles of G are pairwise vertex disjoint (since they are not part of diamonds). We say a triangle or diamond H of G is of *type* i if $|N(V(H)) \backslash V^{\Delta}(G)| = i$. So triangles can be of type i for $i \in \{0, 1, 2, 3\}$, and diamonds can be of type i for $i \in \{0, 1, 2\}$. Let $T_i(G)$ ($D_i(G)$) denote the number of triangles (diamonds) of type i in G. In Figure 1, the numbers next to the triangles and diamonds of G indicate their types.

triangles of G

type i

$T_i(G)$

$D_i(G)$

An edge uv of G is a triangle bridge or *T-bridge* if $u, v \in V^{\Delta}(G)$ but u and v are not part of the same triangle or diamond. The graph $f_1(G)$ is obtained by deleting all vertices of G that are part of type 0 triangles or type 0 diamonds, and in addition deleting all T-bridges. The graph $f_2(G)$ is obtained from G by deleting all vertices in $V^{\Delta}(G)$, so $f_2(G)$ is a subgraph of $f_1(G)$.

T-bridge

$f_1(G)$

$f_2(G)$

Let $cc_1(G) = cc(f_1(G)) - cc(G)$, and let $cc_2(G) = cc(f_2(G)) - cc(f_1(G))$. We will always consider G to be connected, so $cc_1(G) \geq -1$, and this is only an equality when $V^{\Delta}(G) = V(G)$. Since every component of $f_1(G)$ contains a vertex not in $V^{\Delta}(G)$, we have $cc_2(G) \geq 0$. We remark that the graph parameter $x(G)$ that we mentioned in the introduction can now be defined as $x(G) = 2cc_1(G) + cc_2(G) + D_0(G) + T_0(G)$. If the graph in question is clear, we will also write cc_1 and T_0 etc. instead of $cc_1(G)$, $T_0(G)$, etc.

$cc_1(G)$

$cc_2(G)$

4 Constructing and bounding the CD-set

In this section we present our algorithm to construct a minimal 2-CD-set S, and prove an upper bound for the number of triangles in $G[S]$ (Theorem 5), which gives a suitable upper bound for the size of any minimal CD-set $S' \subseteq S$ using Theorem 3. The algorithm is shown in Algorithm 1.

Algorithm 1 An algorithm for finding small CD-sets in cubic graphs

INPUT: A connected, cubic graph G.
OUTPUT: A minimal 2-CD-set S, and a minimal CD-set $S' \subseteq S$.

(Stage 1:)
$S_1 := V(G)$.
while $\exists$ T-bridge uv in $G[S_1]$ s.t. $S_1 - u - v$ is a CD-set of G **do**
    $S_1 := S_1 - u - v$.
(Stage 2:)
$S_2 := S_1$.
repeat
    if $\exists$ type 3 triangle T with $V(T) \subseteq S_2$ s.t. $S_2 \backslash V(T)$ is a CD-set of G **then**
        $S_2 := S_2 \backslash V(T)$.
    if $\exists$ type 2 diamond D with $V(D) \subseteq S_2$ s.t. $S_2 \backslash \textsc{Int}(D)$ is a CD-set of G **then**
        $S_2 := S_2 \backslash \textsc{Int}(D)$.
until no change was made.
(Stage 3:)
$S := S_2$.
Remove vertices (possibly pairwise) from S until it is a minimal 2-CD-set of G.
$S' := S$.
Remove vertices from S' until it is a minimal CD-set of G.

Lemma 4 *Algorithm 1 has a polynomial time implementation, and yields a minimal 2-CD-set S of G and minimal CD-set $S' \subseteq S$ of G.*

Proof: First note that all steps of the algorithm have a polynomial time implementation (for Stage 3 this was observed in Proposition 1). Since in every step either the size of the CD-set under consideration decreases, or the algorithm moves to the next stage, the algorithm terminates in polynomial time.

The changes made to S_1 and S_2 in Stage 1 and 2 always remove sets of adjacent vertices, so these will then have at most two neighbors in the resulting CD-set. So the property of being a 2-CD-set is maintained throughout, and thus S will be a minimal 2-CD-set. □

Theorem 5 *Let S be the minimal 2-CD-set of G constructed by Algorithm 1. Then the number of triangles plus the number of diamonds in $G[S]$ is at most $2cc_1(G) + T_0(G) + D_0(G) + cc_2(G)$.*

Proof: First we analyze Stage 1. Let S_1 denote the set S_1 as it is when Stage 1 has finished. We will show that the number of triangles of type 0, 1 and 2 plus the number of diamonds of type 0 and 1 with all vertices still in S_1 is bounded by $2cc_1 + T_0 + D_0$. In particular, if $V^\Delta = V(G)$, the number of triangles and diamonds with all vertices in S_1 is bounded by $T_0 + D_0 - 2$.

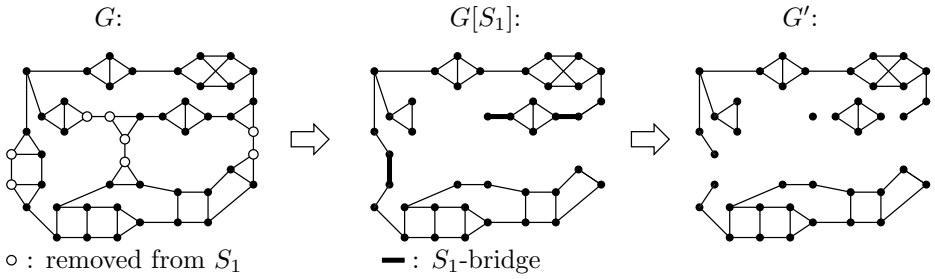

Figure 2: The set S_1 after Stage 1 and G' obtained from $G[S_1]$ by deleting S_1-bridges.

The following definitions are illustrated in Figure 2. A T-bridge uv of G is called an *S_1-bridge* if $u, v \in S_1$. Let $n_B(S_1)$ denote the number of S_1-bridges in $G[S_1]$.

Claim 1 *An S_1-bridge uv is a bridge of $G[S_1]$.*

Claim proof: If uv is not a bridge, then since u and v are both part of triangles or diamonds, removing u and v from S_1 would not disconnect $G[S_1]$. It would only destroy the property that S_1 is a dominating set if all other neighbors of one of the two vertices, say u, are already removed from S_1. But then u is a leaf of $G[S_1]$, so uv is again a bridge. △

Let G' be the subgraph of $G[S_1]$ obtained by removing all S_1-bridges. The components of G' are of two types: those that are part of components of $f_1(G)$, and those that are part of the type 0 triangles and diamonds of G. Hence G' has $1 + cc_1 + T_0 + D_0$ components. (Note that because of the way vertices are removed from S_1 during Stage 1, for every $f_1(G)$ component and every type 0 triangle or diamond, at least one vertex remains in S_1.) Since all S_1-bridges are bridges of $G[S_1]$ (Claim 1), contracting every edge of G' that is not an S_1-bridge gives a tree, so the number of S_1-bridges is

$$n_B(S_1) = cc_1 + T_0 + D_0.$$

Let $T_{i,j}$ ($D_{i,j}$) denote the number of type i triangles (diamonds) of G that contain j vertices of S_1. By counting the number of S_1-bridges that every such triangle or diamond is incident with, we obtain

$$T_{2,3} + 2T_{1,3} + T_{0,1} + 2T_{0,2} + 3T_{0,3} + 2D_{0,4} + D_{0,3} + D_{1,4} \leq 2n_B(S_1).$$

Using the above two inequalities, we can bound the number of type 0, 1, 2 triangles and type 0, 1 diamonds of G that are still fully part of $G[S_1]$:

$$T_{0,3} + T_{1,3} + T_{2,3} + D_{0,4} + D_{1,4} \leq 2n_B(S_1) - 2T_{0,3} - T_{0,1} - 2T_{0,2} - T_{1,3} - D_{0,4} - D_{0,3} =$$

$$2cc_1 + 2T_0 + 2D_0 - 2T_{0,3} - T_{0,1} - 2T_{0,2} - T_{1,3} - D_{0,4} - D_{0,3} =$$

$$2cc_1 + T_0 + D_0 - T_{0,3} - T_{0,2} - T_{1,3} \leq 2cc_1 + T_0 + D_0.$$

Now we will analyze Stage 2. Let S_2 denote the set S_2 as it is when Stage 2 has finished. Type 3 triangles T with $V(T) \subseteq S_2$ are called *S_2-triangles*, and type 2 diamonds D with $V(D) \subseteq S_2$ are called *S_2-diamonds*. We will show that the number of S_2-triangles plus the number of S_2-diamonds is at most cc_2.

The following arguments are illustrated in Figure 3 (which is not based on the graph G that we have used as example earlier). For a component C of $f_1(G)$, the number of components of $f_2(G)$ that are part of C is $cc_2(C)+1$. We will first show that the number of S_2-triangles plus the number of S_2-diamonds in C is at most $cc_2(C)$. In the example of Figure 3, $cc(f_2(C)) = 5$, so $cc_2(C) = 4$, and there are two S_2-triangles and one S_2-diamond.

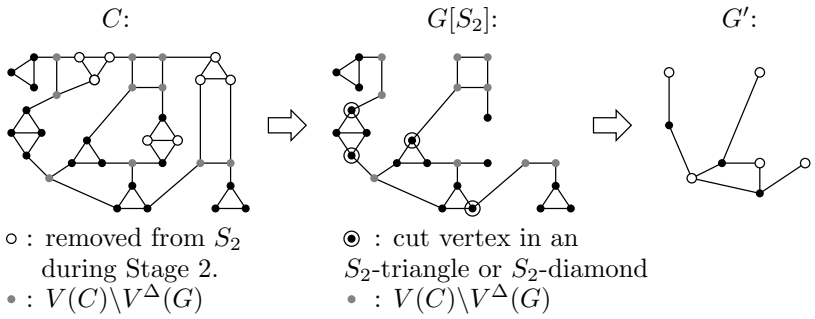

Figure 3: A component C of $f_1(G)$, the set S_2 after Stage 2, and G'.

Consider an S_2-triangle T in C. The set S_2 still contains all vertices of $V(G)\backslash V^{\Delta}$, so since T is of type 3, after removing $V(T)$ from S_2, the set would still be a dominating set of G. Therefore, since $V(T)$ was not removed during Stage 2, $G[S_2\backslash V(T)]$ is not connected. Since T is a triangle and G is cubic, this implies that $V(T)$ contains at least one cut vertex of C. Similarly, for S_2-diamonds it also holds that they contain a cut vertex of C (two actually).

Consider the graph G' defined as follows. For every S_2-triangle and every S_2-diamond in C, we add a black vertex to G', and for every $f_2(G)$ component in C we add a white vertex to G'. When an S_2-triangle or S_2-diamond is adjacent to an $f_2(G)$ component, we add an edge between the corresponding vertices. This gives a bipartite graph G' with $cc_2(C) + 1$ white vertices. Note that since every S_2-triangle and every S_2-diamond contains a cut vertex of C, all black vertices are cut vertices of G'. Using a simple induction argument it then follows that the number of black vertices of G' is at most the number of white vertices minus one, hence is bounded by $cc_2(C)$.

Every S_2-triangle and S_2-diamond of G is part of some component C of $f_1(G)$. In addition, if $\mathcal{C}$ is the set of components of $f_1(G)$, then $cc_2(G) = \sum_{C \in \mathcal{C}} cc_2(C)$. It follows that the number of type 3 triangles plus the number of type 2 diamonds in S_2 is at most $cc_2(G)$.

Since S is a subset of S_1 and of S_2, we know that the number of type 0, 1, 2 triangles plus the number of type 0, 1 diamonds of G that are fully in S is also bounded by $2cc_1 + T_0 + D_0$, and that the number of type 3 triangles plus the number of type 2 diamonds of G that are fully in S is bounded by cc_2. Now all types of triangles and diamonds of G have been considered, which proves the statement. □

The number of triangles and diamonds in $G[S]$ is an upper bound for the number of blocks of $G[S]$ that contain triangles (here we do also mean triangles that are part of diamonds): since S is a minimal 2-CD-set, it is not possible that a diamond of G is not fully part of S but three of its vertices that together form a triangle are. So by combining Lemma 4, Theorem 3 and Theorem 5 we obtain:

Theorem 6 *In polynomial time, Algorithm 1 returns a CD-set S' of G with $|S'| \leq (2n(G) + 2cc_1(G) + cc_2(G) + T_0(G) + D_0(G) - 4)/3$.*

5 An upper bound for the number of leaves

In this section we prove an upper bound for the number of leaves of a spanning tree of a cubic graph. We first observe that we only have to prove this for trees of the following form.

Proposition 7 *A spanning tree T of G with maximum number of leaves exists that contains two edges of every type 0, 1 and 2 triangle of G, and contains either zero or two edges of every type 3 triangle of G.*

Proof: We start with any spanning tree T, and make small changes to T, without decreasing the number of leaves. For all triangles H except those of type 3, we show that we may assume that T contains two edges of H. Let $V(H) = \{v_1, v_2, v_3\}$. First consider the case that T contains only one edge of H, say v_1v_2. Then v_3 is a leaf and at least one of v_1 and v_2, say v_1, is not. Then removing the edge incident with v_3 from T and adding v_1v_3 instead yields again a tree, and no leaves are destroyed. Now consider the case that T contains no edges of H, so all three vertices of H are leaves of T. We assumed H is not of type 3, so H is adjacent to another triangle or diamond H'. W.l.o.g. let $u \in V(H')$ be adjacent to $v_1 \in V(H)$. Since u is part of a triangle or diamond, it can be seen that we may assume that u has degree 2 in T. Now we change T in the following way: we delete uv_1 from T, and the edge of T incident with v_2. We add the edges v_1v_3 and v_2v_3 to T. Now only v_3 loses leaf status, but u becomes a leaf. Both v_1 and v_2 remain leaves. Hence a new spanning tree is obtained with at least as many leaves. Note that this operation does not influence the number of edges of T in other triangles, so we can continue applying such changes until the desired property is achieved. □

Theorem 8 *Let T be a spanning tree of a cubic graph G. Then T has at most $(n(G) - 2cc_1(G) - cc_2(G) - D_0(G) - T_0(G) + 2)/2$ leaves.*

Proof: We will assume T has the properties stated in Proposition 7. The following constructions are illustrated in Figure 4. Let G' be the graph obtained from G by contracting every

diamond and every triangle into a single *black* vertex, and by contracting every component of $f_2(G)$ into a single *white* vertex. Let B (W) denote the set of black (white) vertices of G'. The edges of G that are not contracted, and thus correspond to edges of G' are called *G'-edges* of G. We construct the following spanning subgraph T' of G'. An edge of G' is added to T' if and only if for the corresponding edge $e \in E(G)$: *G'-edges*

- $e \in E(T)$, and
- e is not incident with a leaf of T that is part of a triangle or diamond.

Type 3 triangles that contain three leaves of T will correspond to isolated vertices v of T' at this point. To ensure that T' is connected, in addition we add one arbitrary edge of G' incident with v to T'.

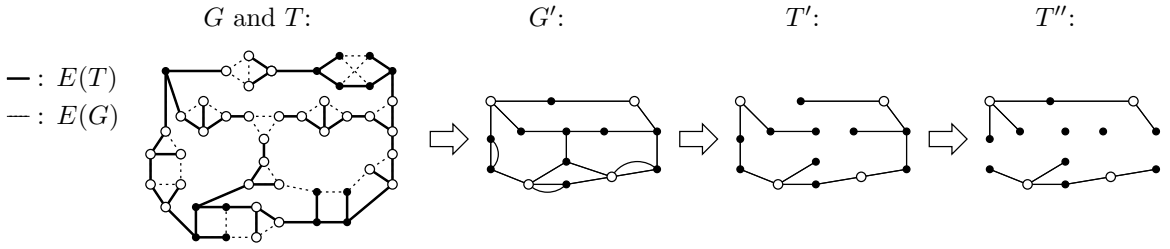

Figure 4: The graphs G, T, G', T' and T'' from the proof of Theorem 8.

Claim 2 *T' is connected.*

Claim proof: Let u and v be two vertices of G', that correspond to subgraphs H_u and H_v of G respectively. We identify a vertex $x_u \in V(H_u)$ as follows. If H_u is a triangle of type 0, 1 or 2, or if H_u is a component of $f_2(G)$, then we may choose $x_u \in V(H_u)$ arbitrarily. If H_u is a triangle of type 3, then for some $w \in V(G')$, we have added uw to $E(T')$. The edge uw corresponds to a G'-edge $e \in E(G)$. Then let x_u be the vertex of H_u that is incident with e. If H_u is a diamond, then we choose x_u to be a vertex of H_u that is not a leaf in T (such a vertex exists). We choose $x_v \in V(H_v)$ using the same rules. Now consider an (x_u, x_v)-path P in T, which exists since T is connected. By construction of T', every G'-edge in P corresponds to an edge that is present in T'. Only for the first and last edge of P this is not trivial, but follows by the choice of x_u and x_v, and because type 0, 1, 2 triangles contain two edges of T (Proposition 7). Hence P corresponds to a path from u to v in T'. △

Claim 3 *If a black vertex v has degree i in T', then the corresponding subgraph H_v of G contains at least $i-1$ vertices that have degree 2 in T.*

Claim proof: If H_v is a triangle, then either T contains two edges of H_v, or H_v is a type 3 triangle that contains three leaves of T (Proposition 7). In the latter case, v has degree 1 in T' and the statement follows trivially. So we may assume H_v contains two edges of T. Since v has degree i in T', there are also at least i G'-edges of G part of T. Considering the few possibilities then yields the statement.

If H_v is a diamond, then we only have to consider the case $i = 2$. By the construction of T', the two vertices of H_v that are incident with the two G'-edges of G can then not be

leaves of T. Then either at least one of them has degree 2 in T, or both have degree 3. But the latter case implies T contains a 4-cycle, a contradiction. $\triangle$

Since T' is connected (Claim 2), we have

$$|E(T')| \geq |V(T')| - 1 = |B| + |W| - 1.$$

Note that all edges of G' are incident with at least one black vertex. Consider the subgraph T'' of T' that has vertex set $V(G')$ again, but only contains those edges of T' that are incident with at least one white vertex. Note that $cc(T'') = D_0(G) + T_0(G) + cc_1 + 1$. When we add the edges of T' one by one until T' is obtained, clearly every edge addition can only decrease the number of components by at most one. Hence $cc(T'') - 1$ is a lower bound for the number of edges of T' that are incident with two black vertices. The degree sum of black vertices is then at least the number of edges of G' plus the number of edges of G' incident with two black vertices. This yields the following bound.

$$\sum_{v \in B} (d_{T'}(v) - 1) \geq |E(T')| + (cc(T'') - 1) - |B| \geq$$

$$|B| + |W| - 1 + D_0 + T_0 + cc_1 - |B| = 2cc_1 + cc_2 + D_0 + T_0.$$

For the last equality we used $|W| = cc_1 + cc_2 + 1$. Since vertices of T' with degree 2 account for at least one vertex of degree 2 in T, and vertices of degree 3 account for at least two such vertices (Claim 3), the above number is also a lower bound for the number of degree 2 vertices in T.

Now let d_i denote the number of vertices of T with degree i, and $n = V(T)$. So $d_1 + d_3 = n - d_2$. For trees with $\Delta \leq 3$ it is easy to see that $d_3 = d_1 - 2$. This yields $2d_1 = n - d_2 + 2 \leq n - (2cc_1 + cc_2 + D_0 + T_0) + 2$, which gives the stated bound. $\square$

6 The approximation guarantee

Theorem 9 *Algorithm 1 is a 4/3-approximation for* MinCD-Set *in cubic graphs, and gives a 3/2-approximation for* MaxLeaf *in cubic graphs.*

Proof: Let $x(G) = 2cc_1(G) + cc_2(G) + T_0(G) + D_0(G)$ and $n = n(G)$. Let S' be the minimal CD-set returned by the algorithm, and let S^* be a *minimum* CD-set of G. By Theorem 6, $|S'| \leq (2n + x(G) - 4)/3 \leq 2(n + x(G) - 2)/3$. By Theorem 8, any spanning tree of G has at most $(n - x(G) + 2)/2$ leaves, so any CD-set of G, in particular S^*, has $|S^*| \geq n - (n - x(G) + 2)/2 = (n + x(G) - 2)/2$. It follows that

$$|S'|/|S^*| \leq \frac{2(n + x(G) - 2)}{3} / \frac{(n + x(G) - 2)}{2} = 4/3.$$

Similarly, using S', a spanning tree T with $l^A \geq n - (2n + x(G) - 4)/3 = (n - x(G) + 4)/3$ leaves can easily be constructed in polynomial time. Since an optimal spanning tree has at most $l^* \leq (n - x(G) + 2)/2 < (n - x(G) + 4)/2$ leaves, the approximation guarantee for MaxLeaf is

$$l^*/l^A < \frac{(n - x(G) + 4)}{2} / \frac{(n - x(G) + 4)}{3} \leq 3/2.$$

$\square$

7 A proof of the bound for minimal 2-CD-sets

In this Section we give a proof of Theorem 3, using notions that were used to prove the similar theorem in [2, 3]. Consider a CD-set S of G. If $S - v$ is not a dominating set of G, then v is called a *dominator* of S. If $G[S - v]$ is not connected, then v is called a *connector* of S. Note that vertices in S may be both dominators and connectors. Observe also that in any *minimal* CD-set S, every vertex is a dominator or connector. Recall that we defined the blocks of a graph G such that bridges of G are not part of blocks of G.

dominator
connector

Proposition 10 *Let S be a minimal 2-CD-set in a cubic graph G with a vertex $v \in S$ that is neither a connector nor a dominator, which is part of block $G[B]$ of $G[S]$. For any CD-set $S' \subseteq S$ with $B \subseteq S'$, v is neither a connector nor a dominator of S'.*

Proof: The set S is a minimal 2-CD-set, so v has three neighbors in S. Since v is not a connector, all three of its neighbors are in B. Because G is cubic, in G the vertex v has no other neighbors other than its three neighbors in B. Therefore, regardless of which vertices are removed from S to obtain S', as long as $B \subseteq S'$ v will not become a dominator or connector. □

The following Lemma is proved in [3] (Lemma 5.10). Here it is formulated only for cubic graphs.

Lemma 11 *Let S be a minimal 2-CD-set of a cubic graph G. For a block H of $G[S]$, let $D_i \subseteq V(H)$ be the vertices that have i neighbors in H ($i \in \{2, 3\}$). Then $|D_2| \geq |D_3|$, and if H contains no triangles, $|D_2| \geq |D_3| + 1$.*

Theorem 3 *Let $G = (V, E)$ be a connected cubic graph. Let S be a minimal 2-CD-set of G where $G[S]$ has b^{Δ} blocks that contain triangles, and let $S' \subseteq S$ be a minimal CD-set of G. Then $|S'| \leq (2n(G) + b^{\Delta}(S') - 4)/3$.*

Proof: Let S be a minimal 2-CD-set of G, and let $S' \subseteq S$ be a minimal CD-set of G. Let L_i denote the vertices in $\overline{S}$ that have i neighbors in S ($i = 1, 2$). and let $L_3 = S \backslash S'$. So we have

$$\overline{S'} = L_1 \cup L_2 \cup L_3.$$

Figure 5 illustrates the following definitions. Consider the graph G' obtained from G by deleting L_2, and deleting all edges that are only incident with vertices in L_1. Consider the tree T obtained from G' by contracting blocks into single vertices. By $V_C \subset V(T)$ we denote the vertices of T resulting from these block contractions. We partition S' into the sets X, N and V_B defined as follows:

- X : vertices v of G' such that $G' - v$ has three components.
- N : vertices of degree 2 in G'.
- V_B: vertices that are part of a block of G', excluding vertices in L_3.

Observe that this indeed yields a partition of S'. For every block $G'[B]$ of G', we partition B into D_2 and D_3, the vertices of degree 2 and 3 respectively in $G'[B]$. Let $I^{\Delta}(B) = 1$ when $G'[B]$ contains a triangle, and $I^{\Delta}(B) = 0$ otherwise. Now we have $|D_2| \geq |D_3| + 1 - I^{\Delta}(B)$

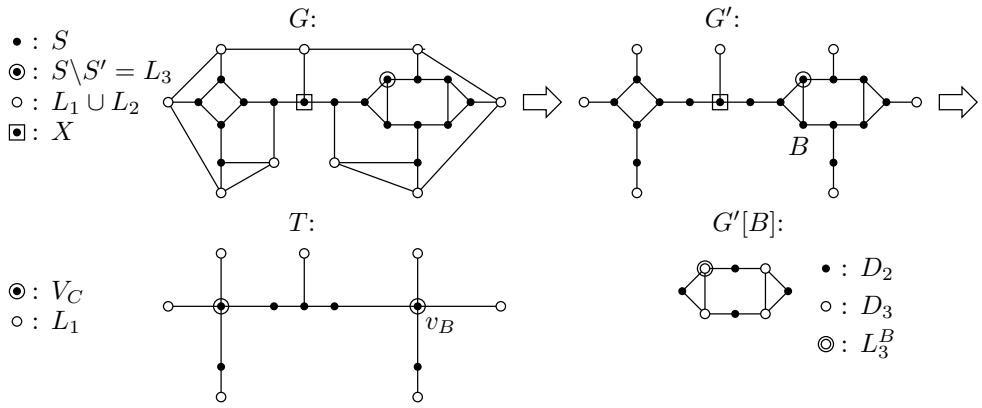

Figure 5: A graph G with a minimal 2-CD-set S, and the resulting graphs G' and T.

(Lemma 11). Let $L_3^B = L_3 \cap B$. Since G is cubic, a vertex in D_3 cannot be a connector or a dominator. So if $D_3 \neq \emptyset$, then B contains at least one vertex of L_3 (Proposition 10), and therefore $|L_3^B| \geq 1$. On the other hand, if $D_3 = \emptyset$ then $G'[B]$ must be a cycle, so we have $|D_2| \geq 4 - I^\Delta(B)$. In both cases the following inequality holds.

$$3|L_3^B| + |D_2| \geq |D_3| + 4 - I^\Delta(B) \Rightarrow 3|L_3^B| + 2|D_2| \geq |D_2| + |D_3| + 4 - I^\Delta(B).$$

Now let $v_B \in V_C$ be the vertex of T corresponding to B. Since S is a minimal 2-CD-set, a vertex in D_2 has a neighbor in G' that is not in the block $G'[B]$, so $d_T(v_B) \geq |D_2|$. Then the above inequality yields

$$3|L_3^B| + 2d_T(v_B) \geq |B| + 4 - I^\Delta(B) \Rightarrow$$

$$2|L_3^B| + 2(d_T(v_B) - 2) \geq |B \backslash L_3^B| - I^\Delta(B).$$

Summing this inequality for all blocks of G' we obtain

$$2|L_3| + 2\sum_{v \in V_C}(d_T(v) - 2) \geq |V_B| - b^\Delta \Rightarrow |L_3| \geq |V_B|/2 - \sum_{v \in V_C}(d_T(v) - 2) - b^\Delta/2.$$

Since vertices in N have degree 3 in G, every such vertex must be adjacent to a vertex in L_2. Vertices in L_2 have at most two neighbors in N, so $|L_2| \geq |N|/2$. Since T is a tree, we know that $\sum_{v \in V(T)} d(v) = 2n(T) - 2$, which gives $|L_1| \geq \sum_{v \in V(T) \backslash L_1}(d_T(v) - 2) + 2$. We can combine these lower bounds to obtain a lower bound for $|\overline{S'}|$:

$$|\overline{S'}| = |L_3| + |L_2| + |L_1| \geq$$

$$|V_B|/2 - \sum_{v \in V_C}(d_T(v) - 2) - b^\Delta/2 \quad + \quad |N|/2 \quad + \sum_{v \in V(T) \backslash L_1}(d_T(v) - 2) + 2 =$$

$$|V_B|/2 - b^\Delta/2 + |N|/2 + \sum_{v \in V(T) \backslash L_1 \backslash V_C}(d_T(v) - 2) + 2.$$

We know that $V(T) \backslash L_1 \backslash V_C = X \cup N$, and vertices in N have degree 2 in T, so we may now write:

$$|\overline{S'}| \geq |V_B|/2 - b^\Delta/2 + |N|/2 + |X| + 2 =$$

$$|S'|/2 - b^{\Delta}/2 + |X|/2 + 2 \;\geq\; |S'|/2 - b^{\Delta}/2 + 2.$$

Here we used $|S'| = |V_B| + |N| + |X|$. Adding $|S'|$ to both sides yields

$$n(G) \geq \frac{3}{2}|S'| - b^{\Delta}/2 + 2 \Rightarrow |S'| \leq (2n(G) + b^{\Delta} - 4)/3.$$

□

References

[1] N. Alon, F. V. Fomin, G. Gutin, M. Krivelevich, and S. Saurabh, *Parameterized algorithms for directed maximum leaf problems*, in ICALP 2007, vol. 4596 of LNCS, Springer, 2007, pp. 352–362.

[2] P. Bonsma, *Spanning trees with many leaves in graphs with minimum degree three*. To appear in SIAM J. Discrete Math.

[3] ——, *Sparse cuts, matching-cuts and leafy trees in graphs*, PhD thesis, University of Twente, Enschede, the Netherlands, 2006. http://purl.org/utwente/57117.

[4] P. Bonsma and F. Zickfeld, *Spanning trees with many leaves in graphs without diamonds and blossoms*. To appear in LATIN 2008.

[5] X. Cheng, X. Huang, D. Li, W. Wu, and D. Du, *A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks*, Networks, 42 (2003), pp. 202–208.

[6] J. R. Correa, C. Fernandes, M. Matamala, and Y. Wakabayashi, *A 5/3-approximation for finding spanning trees with many leaves in cubic graphs*, in WAOA 2007, vol. 4927 of LNCS, Springer, 2008, pp. 184–192.

[7] R. Diestel, *Graph Theory*, Springer-Verlag, New York, 1997.

[8] V. Estivill-Castro, M. R. Fellows, M. A. Langston, and F. A. Rosamond, *FPT is P-time extremal structure I*, in ACiD 2005, vol. 4 of Texts in algorithmics, King's College Publications, 2005, pp. 1–41.

[9] F. V. Fomin, F. Grandoni, and D. Kratsch, *Solving connected dominating set faster than* 2^n, in FSTTCS 2006, vol. 4337 of LNCS, Springer, Berlin, 2006, pp. 152–163.

[10] G. Galbiati, A. Morzenti, and F. Maffioli, *On the approximability of some maximum spanning tree problems*, Theoret. Comput. Sci., 181 (1997), pp. 107–118. LATIN 1995.

[11] M. R. Garey and D. S. Johnson, *Computers and intractability*, Freeman, San Francisco, 1979.

[12] J. R. Griggs, D. J. Kleitman, and A. Shastri, *Spanning trees with many leaves in cubic graphs*, J. Graph Theory, 13 (1989), pp. 669–695.

[13] J. R. Griggs and M. Wu, *Spanning trees in graphs of minimum degree* 4 *or* 5, Discrete Math., 104 (1992), pp. 167–183.

[14] S. Guha and S. Khuller, *Approximation algorithms for connected dominating sets*, Algorithmica, 20 (1998), pp. 374–387.

[15] D. J. Kleitman and D. B. West, *Spanning trees with many leaves*, SIAM J. Discrete Math., 4 (1991), pp. 99–106.

[16] P. Lemke, *The maximum-leaf spanning tree problem in cubic graphs is NP-complete*, IMA publication no. 428, University of Minnesota, Mineapolis, 1988.

[17] K. Loryś and G. Zwoźniak, *Approximation algorithm for the maximum leaf spanning tree problem for cubic graphs*, in Algorithms—ESA 2002, vol. 2461 of LNCS, Springer, Berlin, 2002, pp. 686–697.

[18] H. Lu and R. Ravi, *Approximating maximum leaf spanning trees in almost linear time*, Journal of Algorithms, 29 (1998), pp. 132–141.

[19] L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, and K. Ko, *A greedy approximation for minimum connected dominating sets*, Theoret. Comput. Sci., 329 (2004), pp. 325–330.

[20] R. Solis-Oba, 2-*approximation algorithm for finding a spanning tree with maximum number of leaves*, in Algorithms—ESA 1998, vol. 1461 of LNCS, Springer, Berlin, 1998, pp. 441–452.

[21] F. Zickfeld, *Geometric and combinatorial structures on graphs*, PhD thesis, Technische Universität Berlin, Berlin, 2007.