A $9k$ Kernel for Nonseparating Independent Set in Planar Graphs*

Łukasz Kowalik and Marcin Mucha

Institute of Informatics, University of Warsaw, Poland
{kowalik,mucha}@mimuw.edu.pl

Abstract. We study kernelization (a kind of efficient preprocessing) for NP-hard problems on planar graphs. Our main result is a kernel of size at most $9k$ vertices for the PLANAR MAXIMUM NONSEPARATING INDEPENDENT SET problem. A direct consequence of this result is that PLANAR CONNECTED VERTEX COVER has no kernel with at most $9/8k$ vertices, assuming $\mathbf{P} \neq \mathbf{NP}$. We also show a very simple $5k$-vertices kernel for PLANAR MAX LEAF, which results in a lower bound of $5/4k$ vertices for the kernel of PLANAR CONNECTED DOMINATING SET (also under $\mathbf{P} \neq \mathbf{NP}$).

1 Introduction

Many NP-complete problems, while most likely not solvable efficiently, admit kernelization algorithms, i.e. efficient algorithms which replace input instances with an equivalent, but often much smaller one. More precisely, a *kernelization algorithm* takes an instance I of size n and a parameter $k \in \mathbb{N}$, and after time polynomial in n it outputs an instance I' (called a *kernel*) with a parameter k' such that I is a yes-instance iff I' is a yes instance, $k' \leq k$, and $|I'| \leq f(k)$ for some function f depending only on k. The most desired case is when the function f is polynomial, or even linear (then we say that the problem admits a polynomial or linear kernel). In such a case, when the parameter k is relatively small, the input instance, possibly very large, is "reduced" to a small one. In this paper by the size of the instance $|I|$ we always mean the number of vertices.

In the area of kernelization of graph problems the class of planar graphs (and more generally H-minor-free graphs) is given special attention. This is not only because planar graphs are models of many real-life networks but also because many problems do not admit a (polynomial) kernel for general graphs, while restricted to planar graphs they have a polynomial (usually even linear) kernel. A classic example is the $335k$-vertex kernel for the PLANAR DOMINATING SET due to Alber et al. [1]. In search for optimal results, and motivated by practical applications, recently researchers try to optimize the constants in the linear function bounding the kernel size, e.g. the current best bound for the size of the kernel for the DOMINATING SET is $67k$ [2]. Such improvements often require nontrivial auxiliary combinatorial results which might be of independent interest.

* Work supported by the National Science Centre (grant N N206 567140).

M.C. Golumbic et al. (Eds.): WG 2012, LNCS 7551, pp. 160–171, 2012.

Our paper fits into this framework. We focus on kernelization of the following problem:

MAXIMUM NONSEPARATING INDEPENDENT SET (NSIS) **Parameter:** k
Input: a graph $G = (V, E)$ and an integer $k \in \mathbb{N}$
Question: Is there an independent set I of size at least k such that $G[V - I]$ is connected?

In what follows, $|V|$ is denoted by n. This problem is closely related with CONNECTED VERTEX COVER (CVC in short), where given a graph $G = (V, E)$ and an integer k we ask whether there is a set $S \subseteq V$ of size at most k such that S is a vertex cover (i.e. S touches every edge of G) and S induces a connected subgraph of G. The CVC problem has been intensively studied, in particular there is a series of results on kernels for planar graphs [6,10] culminating in the recent $\frac{11}{3}k$ kernel [8]. It is easy to see that C is a connected vertex cover iff $V - C$ is a nonseparating independent set. In other words, (G, k) is a yes-instance of CVC iff $(G, n - k)$ is a yes-instance of NSIS. In such a case we say that NSIS is a *parametric dual* of CVC. An important property of a parametric dual, discovered by Chen et al [2], is that if the dual problem admits a kernel of size at most αk, then the original problem has no kernel of size smaller than $\alpha/(\alpha - 1)k$, unless P=NP.

As we will see, the NSIS problem in planar graphs is strongly related to the MAX LEAF problem: given a graph G and an integer k, find a spanning tree with at least k leaves.

Our Kernelization Results. We study PLANAR MAXIMUM NONSEPARATING INDEPENDENT SET (PLANAR NSIS in short), which is the NSIS problem restricted to planar graphs. We show a kernel of size at most $9k$ for PLANAR NSIS. This implies that PLANAR CONNECTED VERTEX COVER has no kernel of size smaller than $9/8k$, unless P=NP. This is the first non-trivial lower bound for the kernel size of the PLANAR CVC problem. Our kernelization algorithm is very efficient: it can be implemented to run in $O(n)$ time. As a by-product of our considerations we also show a $5k$ kernel for both MAX LEAF and PLANAR MAX LEAF, which in turn implies a lower bound of $5/4k$ for its parametric dual, i.e. PLANAR CONNECTED DOMINATING SET.

Our Combinatorial Results. Some of our auxiliary combinatorial results might be of independent interest. We mention two of them here. Kleitman and West [7] showed that an n-vertex graph of minimum degree three contains a spanning tree with at least $n/4$ leaves. We generalize their result to graphs that contain no separator consisting of only degree two vertices. We also show that every n-vertex outerplanar graph contains an independent set I and a collection of vertex-disjoint cycles $\mathcal{C}$ such that $9|I| \geq 4n - 3|\mathcal{C}|$.

Previous Results. As the CVC is NP-complete even in planar graphs [5], so is NSIS. To the best of our knowledge there is no prior work on the parameterized complexity of MAXIMUM NONSEPARATING INDEPENDENT SET. The reason for that is simple: a trivial reduction from INDEPENDENT SET (add a vertex connected to all the vertices of the original graph) shows that NSIS is W[1]-hard, i.e. existence of an algorithm of complexity $O(f(k) \cdot |V|^{O(1)})$ is very unlikely (and so is the existence of a polynomial kernel). However, by general results on kernelization for sparse graphs [4], one can see that NSIS admits a $O(k)$ kernel for apex-minor-free graphs, so in particular for planar graphs. However, the general approach does not provide a good bound on the constant hidden in the asymptotic notation. Observe that this constant is crucial: since we deal with an NP-complete problem, in order to find an exact solution in the reduced instance, most likely we need exponential time (or at least superpolynomial, because for planar graphs $2^{O(\sqrt{k})}$-time algorithms are often possible), and the constant appears in the exponent.

Max Leaf has been intensively studied. Although there is a $3.75k$ kernel even for general graphs due to Estivill-Castro et al. [3], some of their reductions do not preserve planarity. Moreover, the algorithm and its analysis are extremely complicated, while our method is rather straightforward.

Yet Another Equivalent Formulation of NSIS. Consider the NSIS problem again. It is easy to see that if graph G has two nontrivial (i.e. with at least two vertices) connected components, then the answer is NO. Furthermore, an instance (G, k) consisting of a connected component C and an independent set I is equivalent to the instance $(G[C], k - |I|)$. Hence, w.l.o.g. we may assume that the input graph G is connected. It is easy to see that the Maximum Nonseparating Independent Set problem for connected graphs is equivalent to the following problem, which we name Maximum Independent Leaf Spanning Tree:

> Maximum Independent Leaf Spanning Tree **Parameter:** k
> **Input:** a graph $G = (V, E)$ and an integer $k \in \mathbb{N}$
> **Question:** Is there a spanning tree T such that the set of leaves of T contains a subset of size k that is independent in G?

In what follows, we will use the above formulation, since it directly corresponds to our approach.

Terminology and Notation. By $N_G(v)$ we denote the set of neighbors of vertex v, $G[S]$ denotes the subgraph of graph G induced by a set of vertices S. If $G = (V, E)$ is a connected graph and $S \subset V$, then we say that S is a separator if $G[V - S]$ is disconnected. By a d-vertex we mean a vertex of degree d.

2 A Simple 12k Kernel for Planar NSIS

In this section we describe a relatively simple algorithm that finds a $12k$ kernel for Planar NSIS. This is achieved by the following three steps. First, in Section 2.1 we show a reduction rule and a linear-time algorithm which, given an instance (G, k), returns an equivalent instance (G', k') such that $|V(G')| \leq |V(G)|$, $k' \leq k$, and moreover G' has no separator consisting of only 2-vertices. Second, in Section 2.2 we show that G' has a spanning tree T with at least $|V(G')|/4$ leaves (and it can be found in linear time). Denote the set of leaves of T by L. Third, in Section 2.3 we show that the graph $G'[L]$ is outerplanar. It follows that $G'[L]$ has an independent set of size at least $|L|/3$ (which can be easily found in linear time) and, consequently, T has at least $|V(G')|/12$ leaves that form an independent set. Hence, if $k' \leq |V(G')|/12$ our algorithm returns the answer YES (and the relevant feasible solution if needed). Otherwise $|V(G')| < 12k' \leq 12k$ so (G', k') is indeed the desired kernel.

2.1 The Separator Rule

Now we describe our main reduction rule, which we call *separator rule*. It is easier for us to prove the correctness of the rule for the CVC problem and then convert it to a rule for the NSIS problem.

> **Separator Rule.** Assume there is a separator S consisting of only 2-vertices. As long as S contains two adjacent vertices, remove one of them from S (note that S is still a separator). Next, choose any $v \in S$ such that the two neighbors a, b of v belong to distinct connected components of $G[V - S]$. If $\deg(a) = \deg(b) = 1$, remove a from G. If $\deg(a) = 1$ and $\deg(b) \geq 2$, remove a from G and decrease the parameter k by 1. Proceed analogously when $\deg(b) = 1$ and $\deg(a) \geq 2$. Finally, when $\deg(a), \deg(b) \geq 2$, contract the path avb into a single vertex v' and decrease k by 2.

We say that a reduction rule for a parameterized problem P is *correct* when for every instance (G, k) of P it returns an instance (G', k') such that:

a) (G', k') is an instance of P,
b) (G, k) is a yes-instance of P iff (G', k') is a yes-instance of P,
c) $k' \leq k$.

Lemma 1. *The separator rule is correct for* Planar CVC.

Proof. Since the separator rule modifies the graph by removing a vertex or contracting a path it is planarity preserving, so a) holds. The condition c) is easy to check so we focus on b), i.e. the equivalence of the instances.

The case when $\deg(a) = \deg(b) = 1$ are trivial so we skip the argument.

Now assume $\deg(a) = 1$, $\deg(b) \geq 2$ (the case $\deg(b) = 1$, $\deg(a) \geq 2$ is symmetric). If C is a minimum connected vertex cover of G, $|C| \leq k$, then $v \in C$ and $a \notin C$. Since $G[C]$ is connected, $\deg(v) = 2$ and $\deg(b) \geq 2$, also $b \in C$. It follows that $C \setminus \{v\}$ is a connected vertex cover of G' of size at most $k' = k - 1$. In the other direction, if C' is a connected vertex cover of G', $|C'| \leq k' = k - 1$, then $b \in C'$ and clearly $C' \cup \{v\}$ is a connected vertex cover of G.

Finally, assume $\deg(a), \deg(b) \geq 2$. Let A and B be the connected components of $G[V - S]$ that contain a and b, respectively. Let a_0 (resp. b_0) be any neighbor of a (resp. b) distinct from v (a_0 and b_0 exist since $\deg(a), \deg(b) \geq 2$). Note that $a_0 \in A \cup S$, $b_0 \in B \cup S$ and $a_0, b_0 \in G'$.

Let us first assume that G' has a connected vertex cover C', $|C'| \leq k'$. We show that G' has a connected vertex cover D', $|D'| \leq k'$ such that $v' \in D'$. Then it is easy to check that $C = (D' \setminus \{v'\}) \cup \{a, v, b\}$ is the required connected vertex cover of G, and $|C| = |D'| + 2 \leq k$.

If $v' \in C'$ we just put $D' = C'$ so suppose that $v' \notin C'$. Then $a_0, b_0 \in C'$. Since $G'[C']$ is connected, there is a path P from a_0 to b_0 in $G'[C']$, possibly of length 0. Since $a_0 \in A \cup S$ and $b_0 \in B \cup S$ we infer that P contains a vertex $w \in S \cap C'$. It follows that $D' = C' \setminus \{w\} \cup \{v'\}$ is a connected vertex cover of G' of size at most k'.

Let us now assume that (G, k) has a connected vertex cover C, $|C| \leq k$. If $\{a, v, b\} \subseteq C$, then clearly $C' = (C \setminus \{a, v, b\}) \cup \{v'\}$ is a connected vertex cover of G' with $|C'| \leq k - 2 = k'$. On the other hand, if $\{a, v, b\} \not\subseteq C$, then $G[C \cup \{a, v, b\}]$ contains a cycle (because C is connected). Since S is a separator, this cycle has to contain some $w_1 \in S$ other than v. In this case we claim that $C' = (C \setminus \{w_1\}) \cup \{a, v, b\}$ is a connected vertex cover of G' and $|C'| \leq k'$. It is clear that C' is a connected vertex cover. The size bound follows from the fact, that C has to contain two out of the three vertices $\{a, v, b\}$. □

Now, we convert the separator rule to the **dual separator rule** as follows. Let (G, ℓ) be an instance of (PLANAR) NSIS. Put $k = |V(G)| - \ell$, apply the separator rule to (G, k) and get (G', k'). Put $\ell' = |V(G')| - k'$ and return (G', ℓ').

Corollary 1. *The dual separator rule is correct for* PLANAR NSIS.

Proof. The condition c) is easy to check while Lemma 1 implies a) and b). □

It is clear that the dual separator rule can be implemented in linear time. However, we would like to stress a stronger claim: there is a linear-time algorithm that given a graph G applies the separator rule as long as it is applicable. This algorithm can be sketched as follows. First we remove all 1-vertices that are adjacent to 2-vertices, and we modify k as described in the separator rule. Second, we find all maximal paths that contain 2-vertices only. For every such path, if it contains at least 3 vertices (and hence two of them form a separator), we replace it by a path of two vertices. It is easy to implement these two steps in linear time. Now, every 2-vertex has at most one neighboring 2-vertex. We remove all 2-vertices that do not have neighbors of degree 2 and for each pair of adjacent 2-vertices we remove exactly one of them. Next, we pick a connected component A of the resulting graph and we mark all its vertices. Then we consider all degree 2 neighbors of this component (that has been removed). If such a neighbor v has an unmarked neighbor then it connects A with another component B. We apply the separator rule to the vertex v (in constant time) and we mark v as *processed.* Then we mark all the vertices of B. As a result the components A and B are joined into a new component A. In any case, the vertex v is not considered any more. We continue the procedure as long as the graph gets connected. All the removed 2-vertices that are not marked as processed are put back in the graph. Since every vertex of G is marked at most once, the whole algorithm works in linear time.

2.2 Finding a Spanning Tree with Many Leaves

Kleitman and West [7] showed how to find a spanning tree with at least $n/4$ leaves in a graph of minimum degree 3. In this section we generalize their result by proving the following theorem.

Theorem 1. *Let G be a connected n-vertex graph that does not contain a separator consisting of only 2-vertices. Then G has a spanning tree with at least $n/4$ leaves. Moreover, such a tree can be found in linear time.*

We will slightly modify the approach of Kleitman and West so that vertices of smaller degree are allowed.

First note that it suffices to show a simplified case where G has no edge uv such that both u and v are 2-vertices (and we still assume the nonexistence of a separator consisting of only 2-vertices). Indeed, if the theorem holds for the simplified case, we just remove from G all the edges uv such that both u and v are 2-vertices, call the new graph $\tilde{G}$. Note that $\tilde{G}$ is connected and $\tilde{G}$ does not contain a separator consisting of only 2-vertices (otherwise the old G contains such a separator). Hence we get a spanning tree T of $\tilde{G}$ with at least $|V(\tilde{G})|/4$ leaves by applying the simplified case. However, since $|V(\tilde{G})| = |V(G)|$ and T is also a spanning tree of G, so T is also the required tree for the general claim. Hence in what follows we assume that G has no edge with both endpoints of degree 2.

In order to build a spanning tree T our algorithm begins with a tree consisting of an arbitrarily chosen vertex (called a *root*), and then the spanning tree is built by a sequence of *expansions*. To expand a leaf $v \in T$ means to add the vertices of $N_G(v) \setminus V(T)$ to T and connect them to v in T. Note that in a tree T built from a root by a sequence of expansions, if a vertex in $V(G) - V(T)$ is adjacent with $v \in V(T)$, then v is a leaf.

The order in which the leaves are expanded is important. To describe this order, we introduce three operations (operations O1 and O3 are the same as in [7], but O2 is modified):

- (O1) Applies when there is a leaf $v \in V(T)$ such that $|N_G(v) \setminus V(T)| \geq 2$. Then v is expanded.
- (O2) Applies when there is a leaf $v \in V(T)$ such that $|N_G(v) \setminus V(T)| = 1$ (let $N_G(v) \setminus V(T) = \{x\}$), and moreover $|N_G(x) \setminus V(T)| = 0$ or $|N_G(x) \cap V(T)| \geq 2$. Then v is expanded.
- (O3) Applies when there is a leaf $v \in V(T)$ such that $|N_G(v) \setminus V(T)| = 1$ (let $N_G(v) \setminus V(T) = \{x\}$), and moreover $|N_G(x) \setminus V(T)| \geq 2$. Then v is expanded and afterwards x is expanded.

Now we can describe the algorithm for Theorem 1, which we call GENERIC:

1. choose an arbitrary vertex $r \in V$ and let $T = \{r\}$,
2. apply O1-O3 as long as possible, giving precedence to O1.

We claim that GENERIC returns a spanning tree of G. Assume for a contradiction that at some point the algorithm is able to apply none of O1-O3 but $V(G) \neq V(T)$. Consider any leaf $v \in T$ such that $N_G(v) \not\subseteq V(T)$. Such a leaf exists because $V(G) \neq V(T)$ and T is built by a sequence of expansions . Since O1 does not apply, $|N_G(v) \setminus V(T)| = 1$. Let $N_G(v) \setminus V(T) = \{x\}$. Since O2 does not apply, $N_G(x) \cap V(T) = \{v\}$. Since neither O2 nor O3 apply, $|N_G(x) \setminus V(T)| = 1$. It follows that $\deg_G(x) = 2$. Moreover, since there are no edges between 2-vertices, the neighbor of x outside of T is not a 2-vertex. It follows that $\bigcup_{v \in L(T)} N_G(v) \setminus V(T)$ is a separator consisting of 2-vertices, which is the desired contradiction.

It remains to show that if a spanning tree T was constructed, then it has at least $n/4$ leaves. It can be done exactly as in the work of Kleitman and West [7]. However, we do it in a different way, in order to introduce and get used to some notation that will be used in later sections, where we describe an improved kernel.

We say that a leaf u of T is *dead* if $N_G(u) \setminus V(T) = \emptyset$. Note that after performing O2 there is at least one new dead leaf: if $|N_G(x) \setminus V(T)| = 0$ then x is a dead leaf, and if $|N_G(x) \cap V(T)| \geq 2$ then all of $(N_G(x) \cap V(T)) \setminus \{v\}$ are dead leaves, because of O1 precedence. For any tree $\hat{T}$, by $L(\hat{T})$ we denote the set of leaves of $\hat{T}$.

Let X_i be the set of the inner vertices of T that were expanded by an operation of type Oi. Let X be the set of the inner vertices of T; note that $X = X_1 \cup X_2 \cup X_3$. Since T is rooted, the standard notions of parent and children apply. For a positive integer i, let P_i denote the set of vertices of T with exactly i children.

Since every vertex besides r is a child of some vertex, we have $\sum_{d \geq 1} d|P_d| = n - 1$. Since the set of vertices with one child is equal to $X_2 \cup (X_3 \cap P_1)$ and $|X_3 \cap P_1| = |X_3 \cap P_{\geq 2}|$ it follows that

$$|X_2| + |X_3 \cap P_{\geq 2}| + \sum_{d \geq 2} d|P_d| = n - 1. \quad (1)$$

Since during O2 at least one leaf dies, $|X_2| \leq |L(T)|$. Similarly, since after expanding a vertex from $X_3 \cap P_{\geq 2}$ the cardinality of $L(T)$ increases, $|X_3 \cap P_{\geq 2}| \leq |L(T)| - 1$. Finally, $\sum_{d \geq 2} d|P_d| \leq \sum_{d \geq 2} 2(d-1)|P_d| = 2(|L(T)| - 1)$. After plugging these three bounds to (1) we get $|L(T)| > n/4$, as required. This finishes the proof of Theorem 1.

2.3 Outerplanarity

Lemma 2. *If G is a planar graph and T is a spanning tree of G, then the graph $G[L(T)]$ is outerplanar.*

Proof. Fix a plane embedding of G and consider the induced plane subgraph $G' = G[L(T)]$. Since T is connected, all vertices of $L(T)$ lie on the same face of G'. Therefore G' is outerplanar. □

Corollary 2. *If G is a planar graph and T is a spanning tree of G then there is a subset of leaves of T of size at least $|L(T)|/3$ that is independent in G (and it can be found in linear time).*

Proof. It is well-known that outerplanar graphs are 3-colorable and the 3-coloring can be found in linear time. So, by Lemma 2 we can 3-color $G[L(T)]$ and we choose the largest color class. □

3 A 9*k* Kernel

In this section we present an improved kernel for the Maximum Independent Leaf Spanning Tree problem. Although the analysis is considerably more involved than that of the $12k$ kernel, the algorithm is almost the same. We need only to force a certain order of the operations O1-O3 in step 2. As before, the algorithm always performs the O1 operation if possible (we will refer to this as *the O1 rule*). Moreover, if more than one O1 operation applies then we choose the one which maximizes the number of vertices added to T (we will refer to this as *the largest branching rule*). If there is still more than one such operation applicable then among them we choose the one which expands a vertex that was added to T later than the vertices which would be expanded by other operation (we will refer to this as *the DFS rule*). Similarly, if there are no O1 operations applicable but more than one O2/O3 operations apply, we also use the DFS rule. The algorithm GENERIC with the order of operations described above will be called BRANCHING.

Note that the algorithm BRANCHING is just a special case of GENERIC, so all the claims we proved in Section 2 apply. Let us think where the bottleneck in this analysis is. There are two sources of trouble: first, if there are many O2/O3 operations we get a spanning tree with few leaves: in particular there might be only O2 operations and O3 operations that add just two leaves (consider a cubic graph which can be built by joining a number of diamonds by edges to form a cycle) and we get roughly $n/4$ leaves. Second, if the outerplanar graph $G[L(T)]$ is far from being bipartite (i.e. has many short odd cycles) then we get a small independent set: in particular, when $G[L(T)]$ is a collection of disjoint triangles, the maximum independent set in $G[L(T)]$ is of size exactly $|L(T)|/3$. However, we will show that these two extremes cannot happen simultaneously. More precisely, we prove the following two theorems.

Theorem 2. *Let G be a connected n-vertex graph that does not contain a separator consisting of only 2-vertices. Then G has a spanning tree T such that if $\mathcal{C}$ is a collection of vertex-disjoint cycles in $G[L(T)]$, then*

$$|L(T)| \geq \frac{n + 3|\mathcal{C}|}{4}.$$

Moreover, T can be found in linear time.

Theorem 3. *Every n-vertex outerplanar graph contains*

- *an independent set I, and*
- *a collection of vertex-disjoint cycles $\mathcal{C}$*

such that $9|I| \geq 4n - 3|\mathcal{C}|$.

Note that Theorem 3 is tight, which is easy to see by considering an outerplanar graph consisting of disjoint triangles. From the above two theorems and Lemma 2 we easily get the following corollary.

Corollary 3. *Let G be a connected n-vertex graph that does not contain a separator consisting of only 2-vertices. Then G has a spanning tree T such that $L(T)$ has a subset of size at least $n/9$ which is independent in G.*

By a similar reasoning as in the beginning of Section 2 we get a $9k$-kernel for the Maximum Independent Leaf Spanning Tree problem. In what follows, we prove Theorem 2. Because of the space constraints the proof of Theorem 3 is deferred to the journal version.

Proof of Theorem 2

Note that similarly as in Theorem 1 it suffices to show a simplified case when G has no edge uv such that both u and v are 2-vertices. Indeed, if the theorem holds for the simplified case, as before we create a new graph $\tilde{G}$ by removing from G all the edges with both endpoints of degree 2 and as before $\tilde{G}$ does not contain a separator consisting of only 2-vertices. Then we apply the simplified case and we get a spanning tree T such that for any collection $\mathcal{C}$ of vertex-disjoint cycles in $\tilde{G}[L(T)]$, we have $|L(T)| \geq (|V(\tilde{G})| + 3|\mathcal{C}|)/4$ and T is a spanning tree of G as well. Moreover, no edge of $E(\tilde{G}) \setminus E(G)$ belongs to a cycle in $G[L(T)]$ for otherwise both of its endpoints have degree at least 3 in G. Hence if $\mathcal{C}$ is a collection of vertex-disjoint cycles in $G[L(T)]$ then it is also a collection of vertex-disjoint cycles in $\tilde{G}[L(T)]$, so the desired inequality holds. Hence in what follows we assume that G has no edge with both endpoints of degree 2. Since we proved that in this case the algorithm GENERIC returns a spanning tree, and each execution of BRANCHING is just a special case of an execution of GENERIC we infer that BRANCHING returns a spanning tree of G, which will be denoted by T.

Let $\mathcal{C}$ be an arbitrary collection of vertex-disjoint cycles in $G[L(T)]$. Our general plan for proving the claim of Theorem 2 is to show that if $|\mathcal{C}|$ is large then we have few O2/O3 operations — by (1) this will improve our bound on $|L(T)|$. To be more precise, let us introduce several definitions.

Recall the O2 operation: it adds a single vertex x to T and at least one leaf of T dies. We choose exactly one of these dead leaves and we *assign* it to x. However, if the vertex x dies we always assign x to itself (so if some other leaves die during this operation, they are unassigned). Let L_u be the set of unassigned leaves of T. Clearly, $|X_2| = |L(T)| - |L_u|$. In order to show that there are few O2 operations, we will show that $|L_u|$ is big.

Let $x_1, x_2, \ldots x_{|X|}$ be the inner vertices of T in the order of expanding them (in particular $x_1 = r$). A *run* is a maximal subsequence $x_b, x_{b+1}, \ldots, x_e$ of vertices from $P_{\geq 2}$, i.e. the nodes in T that have at least two children.

Lemma 3. *Vertices of any run $R = x_b, \ldots, x_e$ form a subtree of T rooted at x_b.*

Proof. Assume that a vertex $x \in \{x_{b+1}, \ldots, x_e\}$ has the parent x_p outside the run. Then $p < b$ and x was a leaf in T while x_b was being expanded. Hence, by the definition of a run, $x_{b-1} \in P_1$, and in particular x_{b-1} was expanded by O2 or O3. However, when this operation was performed, it was possible to expand x by O1, a contradiction with the O1 rule. Hence every vertex $x \in \{x_{b+1}, \ldots, x_e\}$ has the parent in the run, which is equivalent to the claim of the lemma. □

In what follows, the subtree from Lemma 3 is denoted by T_R. Moreover, let $\text{ch}(T_R)$ denote the set of children of the leaves of T_R, i.e.

$$\text{ch}(T_R) = \{v \in V(T) \setminus V(T_R) \ : \ T_R \text{ contains the parent of } v\}.$$

We say that a run R *opens* a cycle C in $\mathcal{C}$ if the first vertex of C that was added to T belongs to $\text{ch}(T_R)$. The following lemma shows a relation between cycles in $\mathcal{C}$ and runs.

Lemma 4. *Every cycle in $\mathcal{C}$ is opened by some run.*

Proof. Consider any cycle $C \in \mathcal{C}$ and let v be the first vertex of C that is added to T. Note that v is not added by O2, for otherwise just after adding v to T, v has at least two neighbors and by the O1 rule v would be the next vertex expanded and hence not a leaf of T, a contradiction. It follows that v is added by O1 or O3 and consequently $v \in \text{ch}(T_R)$ for some run R. □

Now we can sketch our idea for bounding the number of O3 operations (#O3). Both after O1 and O3 the cardinality of $L(T)$ increases. Hence, if we fix the number of leaves in the final tree, then if $|X_1|$ is large then #O3 should be small. Since a run contains at most one vertex of $|X_3|$ (e.g. by Lemma 3), it means that a tree T_R with a large number of children contains plenty of vertices from $|X_1|$. We will show that if a run opens many cycles, then indeed $|\text{ch}(T_R)|$ is large. Let $\mathcal{C}_R$ denote the set of cycles in $\mathcal{C}$ opened by R.

Lemma 5. *Let R be a run. For any cycle $C \in \mathcal{C}_R$ one of the following conditions holds:*

(*i*) $|\text{ch}(T_R) \cap V(C)| + |L_u \cap V(C)| \geq 4$, *or*
(*ii*) $|\text{ch}(T_R) \cap V(C)| + |L_u \cap V(C)| = 3$ *and* $|R \cap P_{\geq 3}| \geq 1$.

Proof. Let v_1 be the vertex of C that is added first to the tree T. By the definition of $\mathcal{C}_R$, $v_1 \in \text{ch}(T_R)$. We see that at least one neighbor of v_1, call it v_2, is in $\text{ch}(T_R)$, for otherwise just after expanding the last vertex of R the vertex v_1 can be expanded by O1, while the algorithm chooses O2/O3, a contradiction with the O1 rule.

Let w be the neighbor of v_2 on C that is distinct from v_1. Assume $w \notin \text{ch}(T_R)$. Then just after expanding the last vertex of R we have $N(v_2) \setminus T = \{w\}$, since if $|N(v_2) \setminus T| \geq 2$ then it is possible to expand v_2 by O1. Hence if v_2 is assigned then it is assigned to w. However, then w is added to T by O2 so w dies during this operation (otherwise w is expanded because of the DFS rule so $w \notin L(T)$), and hence w is assigned to w and $v_2 \in L_u$. To conclude, $w \in \text{ch}(T_R)$ or $v_2 \in L_u$.

If we denote by u the neighbor of v_1 on C that is distinct from v_2, by the same argument we get $u \in \text{ch}(T_R)$ or $v_1 \in L_u$.

It follows that (i) holds, unless $u = w$ (i.e. C is a triangle), $v_1, v_2 \notin L_u$ and $w \in \text{ch}(T_R)$. Let us investigate this last case. We see that $|\text{ch}(T_R) \cap V(C)| = 3$. We will show that $|R \cap P_{\geq 3}| \geq 1$. Since $v_2, w \in \text{ch}(T_R)$, they could not be added by O2 and hence v_1 is assigned to a vertex $x \notin V(C)$. Note that x is added to T after v_2 and w. Assume w.l.o.g. that v_2 was added to T before w. We consider two cases. If v_2 was not added to T by expanding the parent of v_1, then the parent p of v_2 has at least three children (otherwise instead of expanding p the algorithm can expands v_1 and add at least three children, a contradiction with the largest branching rule), so $|R \cap P_{\geq 3}| \geq 1$ as required. Finally, if v_2 was added to T by expanding the parent p of v_1, then $p \in P_{\geq 3}$ for otherwise just after expanding p O1 is applicable to v_1 so either v_1 or v_2 is expanded by the DFS rule. This concludes the proof. □

By applying Lemma 5 to all cycles of a single run R we get the following corollary.

Corollary 4. *For any run R that opens at least one cycle,*

$$|\text{ch}(T_R) \cap V(\mathcal{C}_R)| + |L_u \cap V(\mathcal{C}_R)| + |R \cap P_{\geq 3}| \geq 3|\mathcal{C}_R| + 1.$$

Lemma 6. *For any run R,*

$$|L_u \cap V(\mathcal{C}_R)| + \sum_{d \geq 2} (2d-3)|R \cap P_d| - |R \cap X_3| \geq 3|\mathcal{C}_R|. \tag{2}$$

Because of the space limitations the proof of Lemma 6 is deferred to the journal version. Now we are ready to prove the claim of Theorem 2, i.e. that $|L(T)| \geq (n + 3|\mathcal{C}|)/4$.

Let us add $\sum_{d \geq 2}(2d-3)|P_d|$ to both sides of (1):

$$|X_2| + |X_3 \cap P_{\geq 2}| + 3\sum_{d \geq 2}(d-1)|P_d| = n - 1 + \sum_{d \geq 2}(2d-3)|P_d|. \tag{3}$$

Since $|X_2| = |L(T)| - |L_u|$ and $\sum_{d \geq 2}(d-1)|P_d| = |L(T)| - 1$ we get

$$4|L(T)| = n + 2 + |L_u| + \sum_{d \geq 2}(2d-3)|P_d| - |X_3 \cap P_{\geq 2}|. \tag{4}$$

By Lemma 4 after summing (2) over all runs we get

$$|L_u| + \sum_{d \geq 2}(2d-3)|P_d| - |X_3 \cap P_{\geq 2}| \geq 3|\mathcal{C}|. \tag{5}$$

The claim follows immediately after plugging (5) to (4).

4 A Simple 5k Kernel for (Planar) Max Leaf

In this section we show a simple kernelization algorithm for Max Leaf and Planar Max Leaf. Below we describe three simple rules, which preserve planarity.

– $(1, 2)$**-rule** If there is a 1-vertex u adjacent with a 2-vertex v then remove v.
– **Adjacent 2-vertices Rule** Assume that there are two adjacent 2-vertices u and v. If uv is a bridge, contract uv, otherwise remove uv.
– **Trivial Rule** If G consists of a single edge, return YES if $k \leq 2$.

It is quite clear that the above rules are correct for (Planar) Max Leaf (see e.g. [9], Rules 1-3 for a proof). Note that if none of our rules applies to a connected graph G, then every edge of G has an endpoint of degree at least 3.

Theorem 4. *Let G be a connected graph in which every edge has an endpoint of degree at least 3. Then G has a spanning tree with at least $n/5$ leaves.*

The proof of Theorem 4 is deferred to the journal version.

Let G' be the graph obtained from G by applying our three rules as long as one of them applies. By Theorem 4, if $k \leq n/5$ we can return the answer YES. Hence $n < 5k$ and G' is a 5k-kernel for Planar Max Leaf and Max Leaf.

Acknowledgments. We are very grateful for the reviewers for numerous comments. We also thank Michal Debski for helpful discussions.

References

1. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial-time data reduction for dominating set. J. ACM 51(3), 363–384 (2004)
2. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. SIAM J. Comput. 37(4), 1077–1106 (2007)
3. Estivill-Castro, V., Fellows, M.R., Langston, M.A., Rosamond, F.A.: FPT is P-Time Extremal Structure I. In: ACiD 2005, pp. 1–41 (2005)
4. Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: Charikar, M. (ed.) SODA, pp. 503–510. SIAM (2010)
5. Garey, M.R., Johnson, D.S.: The rectilinear steiner tree problem in NP complete. SIAM Journal of Applied Mathematics 32, 826–834 (1977)
6. Guo, J., Niedermeier, R.: Linear Problem Kernels for NP-Hard Problems on Planar Graphs. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 375–386. Springer, Heidelberg (2007)
7. Kleitman, D.J., West, D.B.: Spanning trees with many leaves. SIAM J. Discrete Math. 4(1), 99–106 (1991)
8. Kowalik, L., Pilipczuk, M., Suchan, K.: Towards optimal kernel for connected vertex cover in planar graphs. CoRR abs/1110.1964 (2011)
9. Prieto-Rodriguez, E.: Systematic kernelization in FPT algorithm design. Ph.D. thesis, University of Newcastle (2005)
10. Wang, J., Yang, Y., Guo, J., Chen, J.: Linear Problem Kernels for Planar Graph Problems with Small Distance Property. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 592–603. Springer, Heidelberg (2011)