

Minimum Leaf Out-branching and Related Problems*

Gregory Gutin[†] Igor Razgon[‡] Eun Jung Kim[§]

Abstract

Given a digraph D , the Minimum Leaf Out-Branching problem (MinLOB) is the problem of finding in D an out-branching with the minimum possible number of leaves, i.e., vertices of out-degree 0. We prove that MinLOB is polynomial-time solvable for acyclic digraphs. In general, MinLOB is NP-hard and we consider three parameterizations of MinLOB. We prove that two of them are NP-complete for every value of the parameter, but the third one is fixed-parameter tractable (FPT). The FPT parametrization is as follows: given a digraph D of order n and a positive integral parameter k , check whether D contains an out-branching with at most $n - k$ leaves (and find such an out-branching if it exists). We find a problem kernel of order $O(k^2)$ and construct an algorithm of running time $O(2^{O(k \log k)} + n^6)$, which is an ‘additive’ FPT algorithm. We also consider transformations from two related problems, the minimum path covering and the maximum internal out-tree problems into MinLOB, which imply that some parameterizations of the two problems are FPT as well.

1 Introduction

We say that a subgraph T of a digraph D is an *out-tree* if T is an oriented tree with only one vertex s of in-degree zero (called *the root*). The vertices of T of out-degree zero are called *leaves* and all other vertices *internal vertices*. If T is a spanning out-tree, i.e. $V(T) = V(D)$, then T is called an *out-branching* of D . It is easy to decide whether a digraph contains an out-branching. In fact, the following holds.

Proposition 1.1. [2] *A digraph D has an out-branching rooted at vertex $r \in V(D)$ if and only if D has a unique strong connectivity component S of D without incoming arcs and $r \in S$. One can check whether D has a unique strong*

*Preliminary extended abstract of this paper appeared in the proceedings of AAIM’08 [14].

[†]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, gutin@cs.rhul.ac.uk

[‡]Department of Computer Science, University College Cork, Ireland, i.razgon@cs.ucc.ie

[§]Department of Computer Science, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, eunjung@cs.rhul.ac.uk

connectivity component and find one, if it exists, in time $O(m+n)$, where n and m are the number of vertices and arcs in D , respectively.

Given a digraph D , the *Minimum Leaf Out-Branching* problem (*MinLOB*) is the problem of finding an out-branching with the minimum possible number of leaves in D . We denote this minimum by $\ell_{\min}(D)$ and if D has no out-branching, we write $\ell_{\min}(D) = 0$.

We first study MinLOB restricted to acyclic digraphs (abbreviated *MinLOB-DAG*). MinLOB-DAG was considered in US patent [7], where its application to the area of database systems was described. Demers and Downing [7] also suggested a heuristic approach to MinLOB-DAG. However no argument or assertion has been made to provide the validity of their approach and to investigate its computational complexity. In fact, the heuristic is not always valid and it is rather slow. Using another approach, we give a simple proof in Section 2 that MinLOB-DAG can be solved in polynomial time.

Since MinLOB generalizes the hamiltonian directed path problem, MinLOB is NP-hard. In this paper, we introduce three parameterizations of MinLOB: (a) $\ell_{\min}(D) \leq k$ ($k \geq 1$), (b) $\ell_{\min}(D) \leq n/k$ ($k \geq 2$), (c) $\ell_{\min}(D) \leq n - k$ ($k \geq 1$), where n is the number of vertices in D and k is the parameter. We show that (a) and (b) are NP-complete for every value of the parameter, but (c) is fixed-parameter tractable and has an algorithm of complexity $O(2^{O(k \log k)} + n^6)$. We also show the existence of a quadratic order kernel for the parameterized problem (c). These results are considered in section (3)-(5). The problem (c) was studied by Prieto and Sloper [19, 20] for undirected graphs (i.e. symmetric digraphs), where the authors obtained an FPT algorithm of complexity $O(2^{2.5k \log k} n^{O(1)})$ and a quadratic order kernel.

In the *minimum path covering* problem (*MinPC*), given a digraph D , our aim is to find the minimum number of vertex-disjoint directed paths, $\text{pc}(D)$, covering all vertices of D . Let $\alpha(D)$ denote the maximum number of mutually non-adjacent vertices of D . It is well-known that MinPC is polynomial time solvable for acyclic digraphs, $\text{pc}(D) \leq \alpha(D)$ for every digraph D (the Gallai-Milgram theorem), and $\text{pc}(D) = \alpha(D)$ for every transitive acyclic digraph D (Dilworth's theorem) [2]. In first part of Section 6, we describe a simple transformation from MinPC into MinLOB which implies that the parameterized problem $\text{pc}(D) \leq n - k$ is fixed-parameter tractable, where n is the number of vertices in D and k is the parameter.

Observe that MinLOB can be reformulated as a problem of finding an out-branching with maximum number of internal vertices. However, while the problem of finding an out-tree with minimum number of leaves in a digraph D is trivial (a vertex is an out-tree), the problem of *finding an out-tree with maximum number of internal vertices* (abbreviated *MaxIOT*) is not trivial; in fact, it is NP-hard (as it also generalizes the hamiltonian directed path problem). The second part of Section 6 is devoted to the latter problem. Further research is

discussed in Section 7.

We recall some basic notions of parameterized complexity here, for a more in-depth treatment of the topic we refer the reader to [8, 11, 18].

A parameterized problem Π can be considered as a set of pairs (I, k) where I is the *problem instance* and k (usually an integer) is the *parameter*. Π is called *fixed-parameter tractable (FPT)* if membership of (I, k) in Π can be decided in time $O(f(k)|I|^c)$, where $|I|$ is the size of I , $f(k)$ is a computable function, and c is a constant independent from k and I . Let Π be a parameterized problem. A *reduction R to a problem kernel* (or *kernelization*) is a many-to-one transformation from $(I, k) \in \Pi$ to $(I', k') \in \Pi$, such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi$, (ii) $k' \leq k$ and $|I'| \leq g(k)$ for some function g and (iii) R is computable in time polynomial in $|I|$ and k . In kernelization, an instance (I, k) is reduced to another instance (I', k') , which is called the *problem kernel*; $|I'|$ is the *size* of the kernel. If I' is a graph, the number of vertices in I' is called the *order* of the kernel.

It is easy to see that a decidable parameterized problem is FPT if and only if it admits a kernelization (cf. [11, 18]); however, the problem kernels obtained by this general result have impractically large size. Therefore, one tries to develop kernelizations that yield problem kernels of smaller size. The survey of Guo and Niedermeier [12] on kernelization lists some problem for which polynomial size kernels and exponential size kernels were obtained. Notice that due to kernelization we can obtain an *additive FPT* algorithm, i.e., an algorithm of running time $O(n^{O(1)} + g(k))$, where $g(k)$ is independent of n , which is often significantly faster than its ‘multiplicative’ counterpart.

All digraphs in this paper are finite with no loops or parallel arcs. We use terminology and notation of [2]; in particular, for a digraph D , $V(D)$ and $A(D)$ denote its vertex and arc sets. The symbols n and m will denote the number of vertices and arcs in the digraph under consideration. The *underlying graph* $UG(D)$ of a digraph D is obtained from D by omitting all orientation of arcs and by deleting one edge from each resulting pair of parallel edges. For a digraph D , an *independent set* (*vertex cover*, respectively) of D is an independent set (vertex cover, respectively) of $UG(D)$. We denote the union of in-neighbors (out-neighbors) of vertices of a set $X \subseteq V(D)$ by $N^-(X)$ ($N^+(X)$); let $N(X) = N^+(X) \cup N^-(X)$.

2 MinLOB-DAG

Let D be an acyclic digraph. We may assume that D has a unique vertex, r , of in-degree 0 as, by Proposition 1.1, this is a necessary and sufficient condition for D to have an out-branching. Let B be a bipartite graph with partite sets $X = V(D)$ and $X' = \{x' : x \in V(D) \setminus \{r'\}\}$ and edge set $E(B) = \{xy' : x \in X, y' \in X', xy \in A(D)\}$. Let $m(B)$ denotes the maximum size of a matching in

B .

Lemma 2.1. *We have $\ell_{\min}(D) = |X| - m(B)$.*

Proof. A set N of edges of B is called *nice* if each vertex of X' is incident to exactly one edge in N and N contains an edge incident to r . Let T be an out-branching of D and let $f(T) = \{xy' : xy \in A(T)\}$. We will prove that f is a bijection between all out-branchings of D and all nice edge sets of B . Indeed, if P is an out-branching, then clearly $f(P)$ is a nice edge set. Let N be a nice edge set and let Q be a spanning subdigraph of D constructed as follows: $xy \in A(Q)$ if and only if $xy' \in N$. Since every vertex of X' is incident to exactly one edge of N , we have $d_Q^-(z) = 1$ for each $z \in V(Q) \setminus \{r\}$. Since Q is acyclic with a unique vertex of in-degree 0, Q is connected and, thus, Q is an out-branching. Clearly, $Q = f^{-1}(N)$.

Let T be an out-branching of D and let $B[f(T)]$ be the subgraph of B induced by the set $f(T)$. Observe that the number of leaves in T equals the number $iv(B[f(T)])$ of isolated vertices in $B[f(T)]$. Let N be a nice edge set in B , let $m(N)$ denote the maximum size of a matching in $B[N]$ and let H be a matching in $B[N]$ of size $m(N)$. Let $y' \in X'$ be a vertex of B not incident to an edge of H and let $xy' \in N$. Since H is maximum, x is incident to an edge of H . Thus, $iv(B[N]) = |X| - m(N)$ and $\ell_{\min}(D) = |X| - \max\{m(N) : N \text{ is nice}\}$.

Let M be a maximum matching in B and let M^* be obtained from M by adding to it an edge $uv' \in E(B)$ for each v' not covered by M . Notice that r is covered by M . Indeed, there exists a vertex u such that r is the only in-neighbor of u in D . Hence if r was not covered by M then u' would not be covered by M either, which means we could extend M by ru' , a contradiction. Therefore, M^* covers r and, by definition, every vertex of X' is incident to exactly one edge of M^* . Thus, M^* is nice. Since $m(B) = m(M^*) = \max\{m(N) : N \text{ is nice}\}$, we conclude that $\ell_{\min}(D) = |X| - m(B)$. \square

The correctness of the algorithm MINLEAF below follows from the proof of Lemma 2.1. The algorithm inputs an acyclic digraph D and outputs a minimum leaf out-branching T , if it exists, and 'NO', otherwise.

MINLEAF

1. **if** the number of vertices with in-degree 0 equals 1 **then**
 $r \leftarrow$ the vertex of in-degree 0 **else** return ‘NO’
2. construct the bipartite graph B of D
3. find a maximum matching M in B and set $M^* \leftarrow M$
4. **for** all $y' \in X'$ not covered by M **do**
 $M^* \leftarrow M^* \cup \{\text{an arbitrary edge incident to } y'\}$
5. $A(T) \leftarrow \emptyset$
6. **for** all $xy' \in M^*$ **do** $A(T) \leftarrow A(T) \cup \{xy\}$
7. return T

Let us analyze the computational complexity of MINLEAF. Let n and m be the number of vertices and arcs in D . Each step of MINLEAF takes at most $O(m)$ time except for Step 3. The computation time required to perform Step 3 is the same as that of solving the maximum cardinality matching problem on a bipartite graph. The last problem can be solved in time $O(|V(B)|^{1.5} \sqrt{|E(B)| / \log |V(B)|})$ [1]. Hence, the algorithm requires at most $O(m + n^{1.5} \sqrt{m / \log n})$ time.

Thus, we have the following:

Theorem 2.2. *Let D be an acyclic digraph. Then MINLEAF returns a minimum leaf out-branching if one exists, or returns ‘NO’ otherwise in time $O(m + n^{1.5} \sqrt{m / \log n})$.*

3 Parameterizations of MinLOB

The following is a natural way to parameterize MinLOB.

MinLOB Parameterized Naturally (MinLOB-PN)

Instance: A digraph D .

Parameter: A positive integer k .

Question: Is $\ell_{\min}(D) \leq k$?

Clearly, this problem is NP-complete already for $k = 1$ as MinLOB-PN for $k = 1$ is equivalent to the hamiltonian directed path problem. Let v be an arbitrary vertex of D . Transform D into a new digraph D_k by adding k vertices v_1, v_2, \dots, v_k together with the arcs vv_1, vv_2, \dots, vv_k . Observe that D has a hamiltonian directed path terminating at v if and only if $\ell_{\min}(D_k) \leq k$. Since the problem is NP-complete of checking whether a digraph has a hamiltonian

directed path terminating at a prescribed vertex, we conclude that MinLOB-PN is NP-complete for every fixed k .

Clearly, $\ell_{\min}(D) \leq n - 1$ for every digraph D of order $n > 1$. Consider a different parameterizations of MinLOB.

MinLOB Parameterized Below Guaranteed Value (MinLOB-PBGV)

Instance: A digraph D of order n with $\ell_{\min}(D) > 0$.

Parameter: A positive integer k .

Question: Is $\ell_{\min}(D) \leq n - k$?

Solution: An out-branching B of D with at most $n - k$ leaves or the answer ‘NO’ to the above question.

Note that we consider MinLOB-PBGV as a constructive problem, not just as a decision problem. Later in the paper we will prove that MinLOB-PBGV is fixed-parameter tractable.

The parametrization MinLOB-PBGV is of the type *below a guaranteed value*. Parameterizations above/below a guaranteed value were first considered by Mahajan and Raman [17] for the problems Max-SAT and Max-Cut; such parameterizations have lately gained much attention, cf. [10, 13, 15, 16, 18] (it worth noting that Heggernes, Paul, Telle, and Villanger [16] recently solved the longstanding minimum interval completion problem, which is a parametrization above guaranteed value). For directed graphs there have been only a couple of results on problems parameterized above/below a guaranteed value, see [3, ?].

Let us denote by $\vec{K}_{1,p-1}$ the *star digraph* of order p , i.e., the digraph with vertices $1, 2, \dots, p$ and arcs $12, 13, \dots, 1p$. Our fixed-parameter tractability result for MinLOB-PBGV may lead us to considering the following stronger (than MinLOB-PBGV) parameterizations of MinLOB.

MinLOB Parameterized Strongly Below Guaranteed Value (MinLOB-PSBGV)

Instance: A digraph D of order n with $\ell_{\min}(D) > 0$.

Parameter: An integer $k \geq 2$.

Question: Is $\ell_{\min}(D) \leq n/k$?

Unfortunately, MinLOB-PSBGV is NP-complete for every fixed $k \geq 2$. To prove this consider a digraph D of order n and a digraph H obtained from D by adding to it the star digraph $\vec{K}_{1,p-1}$ on $p = \lfloor n/(k-1) \rfloor$ vertices ($V(D) \cap V(\vec{K}_{1,p-1}) = \emptyset$) and appending an arc from vertex 1 of $\vec{K}_{1,p-1}$ to an arbitrary vertex y of D . Observe that $\ell_{\min}(H) = p - 1 + \ell_{\min}(D, y)$, where $\ell_{\min}(D, y)$ is the minimum possible number of leaves in an out-branching rooted at y , and that $\frac{1}{k}|V(H)| = p + \epsilon$, where $0 \leq \epsilon < 1$. Thus, $\ell_{\min}(H) \leq \frac{1}{k}|V(H)|$ if and only if $\ell_{\min}(D, y) = 1$. Hence, the hamiltonian directed path problem with fixed initial vertex (vertex y in D) can be reduced to MinLOB-PSBGV for every fixed $k \geq 2$ and, therefore, MinLOB-PSBGV is NP-complete for every $k \geq 2$.

4 Quadratic Order Kernel for MinLOB-PBGV

In this section we introduce a reduction rule for the MinLOB-PBGV problem. Using the reduction rule we present a polynomial time algorithm that either yields an out-branching with at most $n - k$ leaves or produces a kernel whose order is bounded by a quadratic function of k .

Let T be an out-branching of a given digraph D and let $(u, v) \in A(D) \setminus A(T)$. We define the *1-change* for (u, v) as the operation to add the arc (u, v) to T and remove the existing arc $(p(v), v)$ from T , where $p(v)$ is the *parent* (i.e. in-neighbor) of v in T . We say an out-branching is *minimal* if no 1-change for an arc of $A(D) \setminus A(T)$ leads to an out-branching with more internal vertices, or equivalently, less leaves. For two vertices x, y , we write $x \leq_T y$ if there is a path from x to y in T and especially when $x \neq y$, we write $x <_T y$. An arc $(y, x) \in A(D) \setminus A(T)$ is *T -backward* if $x <_T y$. The following is a simple observation on a minimal out-branching.

Lemma 4.1. *Let T be an out-branching of D . Then T is minimal if and only if for every arc $(u, v) \in A(D) \setminus A(T)$ which is not T -backward arc, the vertex u is internal or $d^+(p(v)) = 1$.*

Proof. Suppose the 1-change for $(u, v) \in A(D) \setminus A(T)$ yields an out-branching with less leaves. It is easy to see that (u, v) is not T -backward, u is a leaf and $d^+(p(v)) \geq 2$. Conversely if there is an arc $(u, v) \in A(D) \setminus A(T)$ which is not T -backward, u is a leaf and $d^+(p(v)) \geq 2$ then 1-change for (u, v) produces an out-branching in which the number number of leaves is strictly decreased. \square

Lemma 4.2. *Given a digraph D , we can either build a minimal out-branching T with at most $n - k$ leaves or obtain a vertex cover of size at most $2k - 2$ in $O(n^2m)$ time.*

Proof. Let T be a minimal out-branching. If T has at most $n - k$ leaves, we are done. Suppose it is not. We claim that the set $U = \{u \in V(D) : u \text{ is internal in } T\} \cup \{u \in V(D) : u \text{ is a leaf in } T \text{ and } d^+(p(u)) = 1\}$ is a vertex cover of D . Since the set of internal vertices cover all arcs which are not between the leaves, it suffices to show that every arc (u, v) between two leaves u and v is covered by U . The last statement follows from the fact that T is minimal and Lemma 4.1. What remains is to observe that the number of internal vertices is at most $k - 1$ and the number of leaves which is the only child of its parent is at most $k - 1$ as well.

Now we consider the time complexity of the algorithm. The construction of an out-branching T of D takes $O(n + m)$ time. Whether T is minimal can be checked in $O(nm)$ time since for every arc $(u, v) \in A(D) \setminus A(T)$ we test the conditions of Lemma 4.1. Let L be the list of arcs $(u, v) \in A(D) \setminus A(T)$ which violates the minimality of T , i.e. such that u is a leaf and $d^+(p(v)) \geq 2$.

Whenever $L \neq \emptyset$, choose $(u, v) \in L$ and transform T by replacing the arc $(p(v), v)$ by (u, v) . Accordingly we update the list L as follows: (1) erase all arcs whose tail is u , which takes $O(m)$ time (2) erase all arcs whose head is v , which takes $O(m)$ time (3) add to L arcs of the form (x, y) where x is a leaf of the subtree rooted at v and y is a vertex with $d^+(p(y)) \geq 2$ on the unique path from the root of T to $p(v)$. This takes $O(nm)$ time. The validation of the update with (1)-(3) can be easily verified. Since any out-branching has at least one leaf and we decrease the number of leaves of T by 1 at each transformation, after at most n such transformations we obtain an out-branching where no further transformation can be done. This will be our minimal out-branching. When the minimal out-branching has more than $n - k$ leaves, we can construct the vertex cover U as above in $O(n)$ time. \square

It follows from Lemma 4.2 that we can find either an out-branching which certifies a positive answer for the MinLOB-PBGV problem or a vertex cover of D of size at most $2k - 2$. In the second case, we can remove some redundant vertices from the large independent set of size at least $n - (2k - 2)$ and obtain an instance of smaller size. The *crown structure* plays the fundamental role in this reduction.

Definition 4.3. A crown in a graph G is a pair (H, C) , where $H \subseteq V(G)$ and $C \subseteq V(G)$ with $H \cap C = \emptyset$ such that the following conditions hold:

- (a) The set of neighbors of vertices in C is precisely H , i.e. $H = N(C)$,
- (b) $C = C_m \cup C_u$ is an independent set, and
- (c) There is a perfect matching between C_m and H .

A crown structure is a relatively new idea that allows us to have powerful reduction rules. Its applications have been wide and successful, which includes a linear-size kernel for the vertex cover problem [5, 9].

Given a digraph D , let U be a vertex cover of D . Modify U by adding to it the vertex of in-degree 0 if one exists. Let $W = V(D) \setminus U$ and observe that W is an independent set. Finding an out-branching with at most $n - k$ leaves can be reformulated as the problem of finding an out-branching with at least k internal vertices. Herein we define the *internal number* of D as the largest possible number of internal vertices of an out-branching of D .

In order to accommodate a crown structure to MinLOB-PBGV problem we create an auxiliary model which is similar to those considered in [9, 20]. Note that our model is more refined as we deal with directed graphs unlike [9, 20] which consider only undirected graphs. Given a directed graph D with U and W as above, we build the (undirected) bipartite graph B as follows.

- $V(B) = U' \cup W$, where $U' = N^-(W) \cup (U \times U)$.
- $E(B) = \{\{xy, w\} : xy \in U \times U, w \in W, (x, w) \in A(D), (w, y) \in A(D)\} \cup \{\{x, w\} : x \in U, w \in W, (x, w) \in A(D)\}$

Observe that $N^-(W) \subseteq U$ as U is a vertex cover of D and that no vertex of W in B is isolated since every vertex of W is of in-degree at least one in D .

Lemma 4.4. *If B contains a crown $(H, C = C_m \cup C_u)$ with $C \subseteq W$ and $C_u \neq \emptyset$, then the internal number of D equals the internal number of $D - C_u$.*

Proof. We can extend an out-branching T of $D - C_u$ by appending an arc $(x, w) \in A(D)$, where $w \in C_u$ and x is any in-neighbor of w . The attachment of such an arc does not decrease the number of internal vertices of T . This shows that the internal number of D is not smaller than that of $D - C_u$.

Let a crown $(H, C = C_m \cup C_u)$ with $C \subseteq W$ and a perfect matching M between H and C_m are given. We start with the following claim.

Claim 1. Let c_{root} be the root of T . If $c_{root} \in C$, we can modify the perfect matching M into M' between H and $C'_m \subseteq C$ so that $c_{root} \in C'_m$ and $\{ux, c_{root}\} \in M'$ for some pair vertex $ux \in U \times U$.

Proof of Claim 1. Suppose this is not the case. Recall that c_{root} is of in-degree at least 1 since we excluded any vertex of in-degree 0 from W . Let u be an in-neighbor of c_{root} in D and x be a child of c_{root} in T . Note that $\{u, c_{root}\}, \{ux, c_{root}\} \in E(B)$ and thus $u, ux \in H$.

There are two cases and for each case we can obtain a new perfect matching as follows. Firstly if $c_{root} \in C_u$, simply exchange it with a vertex $c \in C_m$ which is matched to the pair vertex ux by M . This exchange is justified since $\{ux, c_{root}\} \in E(B)$. Secondly suppose $c_{root} \in C_m$ but it is matched to a vertex $u \in N^-(W)$. Since $(u, c_{root}), (c_{root}, x) \in A(D)$, we have the pair vertex ux in U' and moreover it is in H . Hence we can find $c \in C_m$ which is matched to the pair vertex ux and by exchanging it with c_{root} we have a new perfect matching. This is possible as we have $\{ux, c_{root}\} \in E(B)$ and $(u, c) \in A(D)$, thus $\{u, c\} \in E(B)$. \square

Due to Claim 1, when $c_{root} \in C$ we may always assume that $c_{root} \in C_m$ and furthermore that $\{ux, c_{root}\} \in M$ for some pair vertex $ux \in (U \times U)$. Notice that x is not necessarily a child of c_{root} in T .

We shall show that the internal number of $D - C_u$ is not smaller than the internal number of D . To see this suppose T is an out-branching of D and consider the subgraph $F = T - C$ obtained from T by deleting the vertices of C . Obviously F is a union of out-trees, say F_1, \dots, F_l . We will add the vertices of C_m and a set of arcs so that we obtain an out-branching of $D - C_u$ with as many internal vertices as in T at the end of this process.

Recalling that $C \subseteq W$ is an independent set, it is straightforward to see any vertex $c \in C$ falls into one of the three types: (a) c is a leaf in T hanging to some vertex of F (b) c is an internal vertex in T which has both a parent and children in F (c) c is the root c_{root} of T and it has at least one in-neighbor in $V(D)$.

Let $c_1, \dots, c_t \in C$ be the vertices that are of type (b) in T . For each c_i , $1 \leq i \leq t$, let $H_i = \{f_p f_q \in U \times U : (f_p, c_i) \in A(T), (c_i, f_q) \in A(T)\}$. We denote $\bigcup_{1 \leq i \leq t} H_i$ by H_{int} . For the vertex $c_{root} \in C$, let $H_{root} = \{f_p x \in U \times U : (f_p, c_{root}) \in A(D) \setminus A(T), (c_{root}, x) \in A(T)\}$. We set $H_{root} = \emptyset$ if $c_{root} \notin C$. Note that both H_{int} and H_{root} belong to H .

The following procedure defines how to construct an out-tree T'' from F . We initialize $T' \leftarrow F$ and $C_{int} \leftarrow \emptyset$.

1. For every $f_p f_q \in H_{int}$
 - 1.1 let H_i be the unique set containing $f_p f_q$.
 - 1.2 let $c_{pq} \in C_m$ be the vertex with $\{f_p f_q, c_{pq}\} \in M$
 - 1.3 $T' \leftarrow T' + c_{pq} + (f_p, c_{pq}) + (c_{pq}, f_q)$.
 - 1.4 $C_{int} \leftarrow C_{int} \cup \{c_{pq}\}$.
2. $T'' \leftarrow T'$.
3. If $c_{root} \notin C$, return T'' .
4. If $c_{root} \notin C_{int}$
 - 4.1 $T'' \leftarrow T'' + c_{root}$.
 - 4.2 for each child x of c_{root} in T , $T'' \leftarrow T'' + (c_{root}, x)$.
 - 4.3 return T'' .
5. Otherwise
 - 5.1 let $f_p f_q \in H_{int}$ be the vertex with $\{f_p f_q, c_{root}\} \in M$.
 - 5.2 let x be the child of c_{root} in T with $x \leq_{T''} f_p$.
 - 5.3 let $c_x \in C_m$ be the vertex with $\{f_p x, c_x\} \in M$.
 - 5.4 $T'' \leftarrow T'' + c_x + (c_x, x)$.
 - 5.5 for each child $y \neq x$ of c_{root} in T (if any)
 - 5.5.1 let $c_y \in C_m$ be the vertex with $\{f_p y, c_y\} \in M$
 - 5.5.2 $T'' \leftarrow T'' + c_y + (f_p, c_y) + (c_y, y)$.
 - 5.6 return T''

Claim 2. Step 1 is valid and T' at step 2 is a union of out-trees.

Proof of Claim 2. For each $f_p f_q \in H_{int}$, the vertex $f_q \in V(F)$ appears as the second element of the pair vertex in H_{int} at most once. The uniqueness of $H_i \ni f_p f_q$ then follows (step 1.1). Moreover by the construction of H_i , $\{f_p f_q, c_i\} \in E(B)$ and thus $f_p f_q \in N(C) = H$, where the last equality follows by the definition of crown. Hence $f_p f_q$ is uniquely matched to a vertex $c_{pq} \in C_m$ by M (step 1.2). Also $\{f_p f_q, c_{pq}\} \in E(B)$ implies $(f_p, c_{pq}), (c_{pq}, f_q) \in A(D)$, which implies that T' can be properly constructed (step 1.3).

Now observe that any second element f_q of a pair vertex $f_p f_q \in H_{int}$ is a root of an out-tree in F . Thus for each component F_q of F , T' contains at most one arc entering into its root. Moreover, $f_p <_{T'} f_q$ if and only if $f_p <_T f_q$, which

means there is no directed cycle in T' . Witnessing that all the other vertices have at most one arc entering into it, we conclude T' at step 2 is a union of out-trees. \square

We claim that the above procedure returns an out-tree T''

Claim 3. Steps 3-5 are valid and T'' is an out-tree.

Proof of Claim 3. First consider the case when T'' is returned at step 3. With Claim 2, it is enough to show that T' is connected. Let two components F_p and F_q in F be connected by c_i in T . Since $c_{root} \notin C$, the vertex c_i is of type (b) and thus there exist $f_p \in F_p$ and the root f_q of F_q such that $(f_p, c_i) \in A(T)$, $(c_i, f_q) \in A(T)$. By the construction of H_{int} , we have $f_p f_q \in H_i \subseteq H_{int}$ and the vertex $c_{pq} \in C_m$ with $\{f_p f_q, c_{pq}\} \in M$ connects F_p and F_q in T' during the performance of step 1. Hence T' is connected.

If T'' is *not* returned at step 3, we have $c_{root} \in C$. It is important to observe that in this case, the roots of the out-trees in T' at step 2 are exactly the children of c_{root} in T . This is because the root of an out-tree in F has an incoming arc in T' if and only if its parent in T is of type (b).

Secondly suppose that T'' is returned at step 4. Then c_{root} does not participate in T' and c_{root} in T'' is of in-degree 0. By the observation in the second paragraph, T'' is an out-tree.

Thirdly suppose that T'' is returned at step 5. In this case c_{root} has been included as an internal vertex to connect two out-trees in step 1, and the arcs (f_p, c_{root}) and (c_{root}, f_q) have been included in T' , where $f_p f_q$ is the pair vertex found in step 5.1. We want to check that c_x and the arc (c_x, x) in line 5.3 can be properly picked up. Indeed, the pair vertex $f_p x$ belongs to $H_{root} \subseteq H$ and there exists a vertex c_x which is matched to the pair $f_p x$. By the construction of B , the arc (c_x, x) exists as well. Hence at the end of step 5.4, T'' is a union of out-trees whose roots are c_x and the children of c_{root} in T other than x .

If $d_T^+(c_{root}) = 1$, T'' consists of a single out-tree whose root is c_x . Else if $d_T^+(c_{root}) \geq 2$, let y be a child of c_{root} in T and $y \neq x$. Since $(f_p, c_{root}), (c_{root}, y) \in A(D)$, we have the pair vertex $f_p y$ in $H_{root} \subseteq H$ and $f_p y$ is uniquely matched to a vertex c_y . The edge $\{f_p y, c_y\}$ implies the existence of the two arcs (f_p, c_y) , (c_y, y) , hence we can perform step 5.5 properly. Since the vertex f_p is contained in the out-tree rooted at $c_x \in C_m$, the addition of these arcs does not create a cycle. As a result we start at the step 5.5 with $|d_T^+(c_{root})|$ out-trees in the beginning and each time we carry out step 5.5.2, the number of out-trees in T'' decreases by 1. Therefore at the end of step 5.5, we end up with a single out-tree T'' rooted at c_x . \square

During the construction of T'' , we added at least one vertex c_{pq} for each internal vertex c_i of type (b) as an internal vertex of T'' . Also we added at least one vertex as the root or an internal vertex of T'' if $c_{root} \in C$. Hence the number of internal vertices in C for T'' is at least as large as the number

of internal vertices in C for T . Therefore what remains is to see that every vertex f of F which is internal in T can be made to remain internal. The only case we need to consider is a vertex $f \in V(F)$ whose children in T are leaves and all belong to C . Suppose f is a leaf in T' . Since $f \in N(C) = H$, we can uniquely determine a vertex $c_f \in C_m$ such that $\{f, c_f\}$ belongs to the perfect matching M . By the construction of T'' in the above argument, the vertex c_f is not contained in T'' for each such vertex $f \in V(F)$ and thus, we may add c_f and an arc (f, c_f) to T'' while keeping T'' as an out-tree. After this procedure each such vertex f is an internal vertex in T'' , and thus T'' has as many internal vertices as T .

For any vertex c of C_m which does not participate in T' constructed so far, we simply add it to T'' with the arc $(f, c) \in A(D)$. Therefore T'' is an out-branching of $D - C_u$ with as many internal vertices as T . This completes the proof. \square

In light of Lemma 4.4, we have a reduction rule below.

Reduction rule 1. Given a digraph D with a vertex cover U of D and $W = V(D) \setminus U$, construct the associated bipartite graph B . If B has a crown $(H, C = C_m \cup C_u)$ with $C_u \neq \emptyset$, remove the vertices of C_u from D .

We need the following theorem to prove our kernelization lemma.

Theorem 4.5. [9] *Any graph G with an independent set I , where $|I| \geq \frac{2n}{3}$, has a crown (H, C) , where $H \subseteq N(I)$, $C \subseteq I$ and $C_u \neq \emptyset$, that can be found in time $O(nm)$ given I .*

Lemma 4.6 (Kernelization Lemma). *Let D be irreducible. If $|V(D)| > 8k^2 + 6k$ then D has an out-branching with at least k internal vertices.*

Proof. Suppose that D is reduced with $|V(D)| > 8k^2 + 6k$, and that D does not have an out-branching with at least k internal vertices. Since the internal number of D is the same as the internal number of the original digraph, we may assume that D has an out-branching T .

For $|U| < 2k$, we have $|W| = |V(D) \setminus U| > 8k^2 + 4k$ and $|U'| = |N^-(W) \cup (U \times U)| < 2k + 4k^2$. Then $|W| \geq \frac{2|V(B)|}{3}$ which means we have a crown $(H, C = C_m \cup C_u)$ of D with $C \subseteq W$ and $C_u \neq \emptyset$ by Theorem 4.5. This is a contradiction to that D is reduced. \square

Proceeding from what has been discussed above, we give a polynomial time algorithm which computes a quadratic order kernel for the MinLOB-PBGV problem.

KERNELIZATION

1. Build an out-branching T rooted at r by depth-first search.
2. Transform T into a minimal out-branching using 1-change.
3. **If** the number of leaves of T is at most $n - k$, return 'YES'.
4. **Otherwise** Reduce by Rule 1 if possible. If this is not possible, return the instance (it is irreducible).

Let T be the new out-branching obtained by the construction in the proof of Lemma 4.4.

Transform T into a minimal out-branching using 1-change.

Go to line 3.

Step 1-3 take $O(n^2m)$ time by Lemma 4.2. At step 4, we can construct the bipartite graph B in time $O(n^3)$, and $V(B)$ and $E(B)$ are bounded by $n + 2k + 4k^2 = O(n^2)$ and $m + 4k^2n = O(n^3)$ respectively. Due to Theorem 4.5, in $O(n^5)$ time we can reduce the instance by Rule 1 or declare the instance irreducible. Since the size of an instance is strictly decreased at each step of the reduction, we conclude that the algorithm KERNELIZATION runs in $O(n^6)$ time.

5 Solving MinLOB-PBGV

In order to achieve a better running time we provide an alternative way of showing the fixed-parameter tractability of the MinLOB-PBGV problem based on the notion of *tree decomposition*.

A *tree decomposition* of an (undirected) graph G is a pair (X, U) where U is a tree whose vertices we will call *nodes* and $X = \{X_i : i \in V(U)\}$ is a collection of subsets of $V(G)$ (called *bags*) such that

1. $\bigcup_{i \in V(U)} X_i = V(G)$,
2. for each edge $\{v, w\} \in E(G)$, there is an $i \in V(U)$ such that $v, w \in X_i$,
and
3. for each $v \in V(G)$ the set of nodes $\{i : v \in X_i\}$ form a subtree of U .

The *width* of a tree decomposition $(\{X_i : i \in V(U)\}, U)$ equals $\max_{i \in V(U)} \{|X_i| - 1\}$. The *treewidth* of a graph G is the minimum width over all tree decompositions of G . We use the notation $\text{tw}(G)$ to denote the treewidth of a graph G .

By a *tree decomposition of a digraph* D we will mean a tree decomposition of the underlying graph $UG(D)$. Also, $\text{tw}(D) = \text{tw}(UG(D))$.

Theorem 5.1. *There is an polynomial time algorithm that, given an instance (D, k) of the MinLOB-PBGV problem, either finds a solution or establishes a tree decomposition of D of width at most $2k - 2$.*

Proof. By Lemma 4.2, there is a polynomial time algorithm which either finds a solution or specifies a vertex cover C of D of size at most $2k - 2$. Let $I = \{v_1, \dots, v_s\} = V(D) \setminus C$. Consider a star U with nodes x_0, x_1, \dots, x_s and edges $x_0x_1, x_0x_2, \dots, x_0x_s$. Let $X_0 = C$ and $X_i = X_0 \cup \{v_i\}$ for $i = 1, 2, \dots, s$ and let X_j be the bag corresponding to x_j for every $j = 0, 1, \dots, s$. Observe that $(\{X_0, X_1, \dots, X_s\}, U)$ is a tree decomposition of D and its width is at most $2k - 2$. \square

Theorem 5.1 shows that an instance (D, k) of the MinLOB-PBGV problem can be reduced to another instance with treewidth $O(k)$. Using standard dynamic programming techniques we can solve this instance in time $2^{O(k \log k)} n^{O(1)}$. We can further accelerate the solution procedure using kernelization. If we first find the kernel and then establish the tree decomposition, the resulting algorithm will run in time $O(2^{O(k \log k)} + n^6)$. Now we have the following result.

Theorem 5.2. *The MinLOB-PBGV problem can be solved by an additive FPT algorithm of running time $O(2^{O(k \log k)} + n^6)$.*

6 Related Problems

In this section we consider transformations from MinPC and MaxIOT introduced in Section 1 into MinLOB. We start from MinPC.

For a digraph D , let $\text{pc}(D)$ be the minimum number of vertex-disjoint directed paths in D . We have the following:

Proposition 6.1. *Let $D = (V, A)$ be a digraph and let \hat{D} be the digraph obtained from D by adding a new vertex s and all possible arcs from s to V . Then $\text{pc}(D) = \ell_{\min}(\hat{D})$.*

Proof. Since a collection of p disjoint directed paths in D covering $V(D)$ corresponds to an out-branching of \hat{D} with p leaves, we have $\text{pc}(D) \geq \ell_{\min}(\hat{D})$. Let B be an out-branching of \hat{D} with p leaves. We say that a vertex x of B is *branching* if $d_B^+(x) > 1$. Consider a maximal directed path Q of B containing a leaf but not containing branching vertices. Observe that $B - V(Q)$ has $p - 1$ leaves. Thus, we can decompose the vertices of B into p disjoint directed paths. Deleting the vertex s from this collection of paths, we see that $\text{pc}(D) \leq \ell_{\min}(\hat{D})$. Thus, $\text{pc}(D) = \ell_{\min}(\hat{D})$. \square

Fixed-parameter tractability of MinLOB-PBGV and Proposition 6.1 imply that the parameterized problem $\text{pc}(D) \leq n - k$ is FPT, too.

For a digraph D and a vertex v in D , let D_v denote the subgraph of D obtained from the subgraph of D induced by all vertices reachable from v by deleting all arcs entering v . The following result allows us to reduce MaxIOT to MinLOB.

Proposition 6.2. *Let D be a digraph and let S be the set of vertices belonging to all strongly connected components of D without incoming arcs. Let B_v be an out-branching of D_v of minimum number of leaves, and let s be a vertex of S such that $\ell_{\min}(B_s) \leq \ell_{\min}(B_v)$ for each $v \in S$. Then B_s is a maximum internal out-tree of D .*

Proof. Let T be a solution to MaxIOT for D with maximum possible number of leaves and let r be the root of T . Observe that $r \in S$ as otherwise we would be able to extend T to an out-tree T' with more internal vertices such that the root of T' is in S . Observe also that T is an out-branching of D_r as otherwise we would be able to extend T to an out-branching T' of D_r such that T' has more either leaves or internal vertices than T . Clearly, $\ell_{\min}(B_r) \leq \ell_{\min}(B_v)$ for each $v \in S$. \square

Together with the above-proved results, Proposition 6.2 implies that MaxIOT for acyclic digraphs is polynomially-time solvable and that the problem of finding an out-tree with at least k internal vertices in an arbitrary digraph D is FPT. Recall that the problem of finding an out-branching with at least k internal vertices has a quadratic order kernel. However, the problem of finding an out-tree with at least k internal vertices does not have a polynomial size kernel unless $\text{PH} = \Sigma_p^3$. This easily follows from Lemmas 1-3 in [4].

7 Further Research

We have proved that MinLOB-PBGV is FPT and suggested an $2^{O(k \log k)} \cdot n^{O(1)}$ algorithm for the problem. Our algorithm uses a treewidth approach. Very recently, using a different approach Cohen et al. [6] designed an $2^{O(k)} \cdot n^{O(1)}$ algorithm for MinLOB-PBGV. An open interesting question is whether MinLOB-PBGV admits a linear order kernel or not.

Acknowledgements Research of Gutin and Kim was supported in part by an EPSRC grant. Part of the paper was written when Razgon was visiting Department of Computer Science, Royal Holloway, University of London. The research of Razgon at the Department of Computer Science, University College Cork was supported by Science Foundation Ireland Grant 05/IN/I886.

References

- [1] Helmut Alt, Norbert Blum, Kurt Mehlhorn, and Markus Paul. Computing a maximum cardinality matching in a bipartite graph in time $o(n^{1.5} \sqrt{m/\log n})$. *Inf. Process. Lett.*, 37(4), 1991.
- [2] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag, 2000.

- [3] Jørgen Bang-Jensen and Anders Yeo. The minimum spanning strong subgraph problem is fixed parameter tractable. *Discrete Applied Mathematics*, 156(15):2924–2929, 2008.
- [4] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels (extended abstract). In *ICALP (1)*, volume 5125 of *Lecture Notes in Computer Science*, pages 563–574, 2008.
- [5] Benny Chor, Mike Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *WG*, volume 3353 of *Lecture Notes in Computer Science*, pages 257–269, 2004.
- [6] Nathan Cohen, Fedor Fomin, Gregory Gutin, Eun Jung Kim, Saket Saurabh, and Anders Yeo, Algorithm for Finding k -Vertex Out-trees and its Application to k -Internal Out-branching Problem. Tech. Report arXiv:0903.0938, March, 2009
- [7] Alan Demers and Alan Downing. Minimum leaf spanning tree. *US Patent no. 6,105,018*, 2008.
- [8] Rod Downey and Mike Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [9] Mike Fellows, Pinar Heggernes, Frances A. Rosamond, Christian Sloper, and Jan Arne Telle. Finding k disjoint triangles in an arbitrary graph. In *WG*, volume 3353 of *Lecture Notes in Computer Science*, pages 235–244, 2004.
- [10] Henning Fernau. Parameterized algorithmics for linear arrangement problems. *Discrete Applied Mathematics*, 156(17):3166–3177, 2008.
- [11] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006.
- [12] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
- [13] Gregory Gutin, Arash Rafiey, Stefan Szeider, and Anders Yeo. The linear arrangement problem parameterized above guaranteed value. *Theory of Computing Systems*, 41(3):521–538, 2007.
- [14] Gregory Gutin, Igor Razgon, and Eun Jung Kim. Minimum leaf out-branching problems. In *AAIM*, volume 5034 of *Lecture Notes in Computer Science*, pages 235–246, 2008.

- [15] Gregory Gutin, Stefan Szeider, and Anders Yeo. Fixed-parameter complexity of minimum profile problems. *Algorithmica*, 52(2):133–152, 2008.
- [16] Pinar Heggeres, Christophe Paul, Jan Arne Telle, and Yngve Villanger. Interval completion with few edges. In *STOC*, pages 374–381, 2007.
- [17] Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *Journal of Algorithms*, 31(2):335–354, 1999.
- [18] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 31. Oxford Lecture Series in Mathematics and Its Applications, 2006.
- [19] Elena Prieto and Christian Sloper. Either/or: Using vertex cover structure in designing fpt-algorithms - the case of k-internal spanning tree. In *WADS*, volume 2748 of *Lecture Notes in Computer Science*, pages 474–483, 2003.
- [20] Elena Prieto and Christian Sloper. Reducing to independent set structure – the case of k-internal spanning tree. *Nordic Journal of Computing*, 12(3):308–318, 2005.