Algorithms for k-Internal Out-Branching

Meirav Zehavi

Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel
meizeh@cs.technion.ac.il

Abstract. The *k-Internal Out-Branching (k-IOB)* problem asks if a given *directed* graph has an *out-branching* (i.e., a spanning tree with exactly one node of in-degree 0) with at least k internal nodes. The *k-Internal Spanning Tree (k-IST)* problem is a special case of k-IOB, which asks if a given *undirected* graph has a spanning tree with at least k internal nodes. We present an $O^*(4^k)$ time randomized algorithm for k-IOB, which improves the O^* running times of the best known algorithms for both k-IOB and k-IST. Moreover, for graphs of bounded degree Δ, we present an $O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})$ time randomized algorithm for k-IOB. Both our algorithms use polynomial space.

1 Introduction

In this paper we study the *k-Internal Out-Branching (k-IOB)* problem. The input for k-IOB consists of a *directed* graph $G = (V, E)$ and a parameter $k \in \mathbb{N}$, and the objective is to decide if G has an *out-branching* (i.e., a spanning tree with exactly one node of in-degree 0, that we call the root) with at least k internal nodes (i.e., nodes of out-degree ≥ 1). The k-IOB problem is of interest in database systems [2].

A special case of k-IOB, called *k-Internal Spanning Tree (k-IST)*, asks if a given *undirected* graph $G = (V, E)$ has a spanning tree with at least k internal nodes. A possible application of k-IST, for connecting cities with water pipes, is given in [14].

The k-IST problem is NP-hard even for graphs of bounded degree 3, since it generalizes the Hamiltonian path problem for such graphs [5]; thus k-IOB is also NP-hard for such graphs. In this paper we present parameterized algorithms for k-IOB. Such algorithms are an approach to solve NP-hard problems by confining the combinatorial explosion to a parameter k. More precisely, a problem is *fixed-parameter tractable (FPT)* with respect to a parameter k if an instance of size n can be solved in $O^*(f(k))$ time for some function f [10].[1]

Related Work: Nederlof [9] gave an $O^*(2^{|V|})$ time and polynomial space algorithm for k-IST. For graphs of bounded degree Δ, Raible et al. [14] gave an $O^*(((2^{\Delta+1} - 1)^{\frac{1}{\Delta+1}})^{|V|})$ time and exponential space algorithm for k-IST.

[1] O^* hides factors polynomial in the input size.

G. Gutin and S. Szeider (Eds.): IPEC 2013, LNCS 8246, pp. 361–373, 2013.
© Springer International Publishing Switzerland 2013

Table 1. Known parameterized algorithms for k-IOB and k-IST

Reference	Variation	Time Complexity	The Topology of G
Priesto et al. [12]	k-IST	$O^*(2^{O(k \log k)})$	General
Gutin al. [6]	k-IOB	$O^*(2^{O(k \log k)})$	General
Cohen et al. [1]	k-IOB	$O^*(49.4^k)$	General
Fomin et al. [4]	k-IOB	$O^*(16^{k+o(k)})$	General
Fomin et al. [3]	k-IST	$O^*(8^k)$	General
Raible et al. [14]	k-IST	$O^*(2.1364^k)$	$\Delta = 3$
This paper	**k-IOB** **k-IOB**	$\mathbf{O^*(4^k)}$ $\mathbf{O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})}$	**General** $\mathbf{\Delta = O(1)}$

Table 2. Some concrete figures for the running time of the algorithm Δ-IOB-Alg

Δ	3	4	5	6
Time complexity	$O^*(2.51985^k)$	$O^*(2.99662^k)$	$O^*(3.24901^k)$	$O^*(3.40267^k)$

Table 1 presents a summary of known parameterized algorithms for k-IOB and k-IST. In particular, the algorithms having the best known O^* running times for k-IOB and k-IST are due to [4], [3] and [14]. Fomin et al. [4] gave an $O^*(16^{k+o(k)})$ time and polynomial space randomized algorithm for k-IOB, and an $O^*(16^{k+o(k)})$ time and $O^*(4^{k+o(k)})$ space deterministic algorithm for k-IOB. Fomin et al. [3] gave an $O^*(8^k)$ time and polynomial space deterministic algorithm for k-IST. For graphs of bounded degree 3, Raible et al. [14] gave an $O^*(2.1364^k)$ time and polynomial space deterministic algorithm for k-IST.

Further information on k-IOB, k-IST and variants of these problems is given in surveys [11,15].

Our Contribution: We present an $O^*(4^k)$ time and polynomial space randomized algorithm for k-IOB, that we call IOB-Alg. Our algorithm IOB-Alg improves the O^* running times of the best known algorithms for both k-IOB and k-IST.

For graphs of bounded degree Δ, we present an $O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})$ time and polynomial space randomized algorithm for k-IOB, that we call Δ-IOB-Alg. Some concrete figures for the running time of Δ-IOB-Alg are given in Table 2.

Techniques: Our algorithm IOB-Alg is based on two reductions as follows. We first reduce k-IOB to a new problem, that we call (k, l)-*Tree*, by using an observation from [1]. This reduction allows us to focus our attention on finding a tree whose size depends on k, rather than a spanning tree whose size depends on $|V|$. We then reduce (k, l)-Tree to the *t-Multilinear Detection (t-MLD)* problem, which concerns multivariate polynomials and has an $O^*(2^t)$ time randomized algorithm [7,17]. We note that reductions to t-MLD have been used to solve several problems quickly (see, e.g., [8]). IOB-Alg is another proof of the applicability of this new tool.

Our algorithm Δ-IOB-Alg, though based on the same technique as IOB-Alg, requires additional new non-trivial ideas and is more technical. In particular, we now use a proper coloring of the graph G when reducing (k, l)-Tree to t-MLD. This idea might be useful in solving other problems.

Organization: Section 2 presents our algorithm IOB-Alg. Specifically, Section 2.1 defines (k, l)-*Tree*, and presents an algorithm that solves k-IOB by using an algorithm for (k, l)-Tree. Section 2.2 defines t-MLD, and reduces (k, l)-Tree to t-MLD. Then, Section 2.3 presents our algorithm for (k, l)-Tree, and thus concludes IOB-Alg. Section 3 presents our algorithm Δ-IOB-Alg. Specifically, Section 3.1 modifies the algorithm presented in Section 2.1, Section 3.2 modifies the reduction presented in Section 2.2, and Section 3.3 modifies the algorithms presented in Section 2.3. Finally, Section 4 presents a few open questions.

2 An $O^*(4^k)$-time k-IOB Algorithm

2.1 The (k, l)-Tree Problem

We first define a new problem, that we call (k, l)-*Tree*.

(k, l)-Tree

- Input: A directed graph $G = (V, E)$, a node $r \in V$, and parameters $k, l \in \mathbb{N}$.
- Goal: Decide if G has an *out-tree* (i.e., a tree with exactly one node of in-degree 0) rooted at r with exactly k internal nodes and l leaves.

We now show that we can focus our attention on solving (k, l)-Tree. Let $\mathsf{A}(G, r, k, l)$ be a $t(G, r, k, l)$ time and $s(G, r, k, l)$ space algorithm for (k, l)-Tree.

Algorithm 1. IOB-Alg[A](G, k)

```
1: for all r ∈ V do
2:    if G has no out-branching T rooted at r then Go to the next iteration. end if
3:    for l = 1, 2, ..., k do
4:       if A(G, r, k, l) accepts then Accept. end if
5:    end for
6: end for
7: Reject.
```

The following observation immediately implies the correctness of IOB-Alg[A] (see Algorithm 1).

Observation 1 ([1]). *Let $G = (V, E)$ be a directed graph, and $r \in V$ such that G has an out-branching rooted at r.*

- *If G has an out-branching rooted at r with at least k internal nodes, then G has an out-tree rooted at r with exactly k internal nodes and at most k leaves.*
- *If G has an out-tree rooted at r with exactly k internal nodes, then G has an out-branching with at least k internal nodes.*

By Observation 1, and since Step 2 can be performed in $O(|E|)$ time and $O(|V|)$ space (e.g., by using DFS), we have the following result.

Lemma 1. IOB-Alg[A] *is an* $O(\sum_{r\in V}(|E|+\sum_{1\leq l\leq k}t(G,r,k,l)))$ *time and* $O(|V|+\max_{r\in V,1\leq l\leq k}s(G,r,k,l))$ *space algorithm for k-IOB.*

2.2 A Reduction from (k,l)-Tree to t-MLD

We first give the definition of t-MLD [7].

t-MLD

- Input: A polynomial P represented by an arithmetic circuit C over a set of variables X, and a parameter $t \in \mathbb{N}$.
- Goal: Decide if P has a multilinear monomial of degree at most t.

Let (G,r,k,l) be an input for (k,l)-Tree. We now construct an input $f(G,r,k,l)=(C_{r,k,l},X,t)$ for t-MLD. We introduce an indeterminate x_v for each $v\in V$, and define $X=\{x_v : v\in V\}$ and $t=k+l$.

The idea behind the construction is to let each monomial represent a pair of an out-tree $T=(V_T,E_T)$ and a function $h: V_T\rightarrow V$, such that if $(v,u)\in E_T$, then $(h(v),h(u))\in E$ (i.e., h is a homomorphism). The monomial is $\prod_{v\in V_T}x_{h(v)}$. We get that the monomial is multilinear iff $\{h(v) : v\in V_T\}$ is a set (then $h(T)=(\{h(v):v\in V_T\},\{(h(v),h(u)):(v,u)\in E_T\})$ is an out-tree).

Towards presenting $C_{r,k,l}$, we inductively define an arithmetic circuit $C_{v,k',l'}$ over X , for all $v\in V, k'\in\{0,...,k\}$ and $l'\in\{1,...,l\}$. Informally, the multilinear monomials of the polynomial represented by $C_{v,k',l'}$ represent out-trees of G rooted at v that have exactly k' internal nodes and l' leaves.

Base Cases:

1. If $k'=0$ and $l'=1$: $C_{v,k',l'}=x_v$.
2. If $k'=0$ and $l'>1$: $C_{v,k',l'}=0$.

Steps:

1. If $k'>0$ and $l'=1$: $C_{v,k',l'}=\sum_{u \text{ s.t.}(v,u)\in E}x_vC_{u,k'-1,l'}$.
2. If $k'>0$ and $l'>1$: $C_{v,k',l'}=$
   $\sum_{u \text{ s.t.}(v,u)\in E}(x_vC_{u,k'-1,l'}+\sum_{1\leq k^*\leq k'}\sum_{1\leq l^*\leq l'-1}C_{v,k^*,l^*}\cdot C_{u,k'-k^*,l'-l^*})$.

The following order shows that when computing an arithmetic circuit $C_{v,k',l'}$, we only use arithmetic circuits that have been already computed.

Order:

1. For $k'=0,1,...,k$:
   (a) For $l'=1,2,...,l$:
      i. $\forall v\in V$: Compute $C_{v,k',l'}$.

Denote the polynomial that $C_{v,k',l'}$ represents by $P_{v,k',l'}$.

Lemma 2. (G,r,k,l) *has a solution iff* $(C_{r,k,l},X,t)$ *has a solution.*

Proof. By using induction, we first prove that if G has an out-tree $T = (V_T, E_T)$ rooted at v with exactly k' internal nodes and l' leaves, then $P_{v,k',l'}$ has the (multilinear) monomial $\prod_{w \in V_T} x_w$.

The claim is clearly true for the base cases, and thus we next assume that $k' > 0$, and the claim is true for all $v \in V$, $k^* \in \{0, ..., k'\}$ and $l^* \in \{1, ..., l'\}$, such that ($k^* < k'$ or $l^* < l'$).

Let $T = (V_T, E_T)$ be an out-tree of G, that is rooted at v and has exactly k' internal nodes and l' leaves. Also, let u be a neighbor of v in T. Denote by $T_v = (V_v, E_v)$ and $T_u = (V_u, E_u)$ the two out-trees of G in the forest $F = (V_T, E_T \setminus \{(v, u)\})$, such that $v \in V_v$. We have the following cases.

1. If $|V_v| = 1$: T_u has $k' - 1$ internal nodes and l' leaves. By the induction hypothesis, $P_{u,k'-1,l'}$ has the monomial $\prod_{w \in V_u} x_w$. Thus, by the definition of $C_{v,k',l'}$, $P_{v,k',l'}$ has the monomial $x_v \prod_{w \in V_u} x_w = \prod_{w \in V_T} x_w$.
2. Else: Denote the number of internal nodes and leaves in T_v by k_v and l_v, respectively. By the induction hypothesis, P_{v,k_v,l_v} has the monomial $\prod_{w \in V_v} x_w$, and $P_{u,k'-k_v,l'-l_v}$ has the monomial $\prod_{w \in V_u} x_w$. By the definition of $C_{v,k',l'}$, $P_{v,k',l'}$ has the monomial $\prod_{w \in V_v} x_w \prod_{w \in V_u} x_w = \prod_{w \in V_T} x_w$.

Now, by using induction, we prove that if $P_{v,k',l'}$ has the (multilinear) monomial $\prod_{w \in U} x_w$, for some $U \subseteq V$, then G has an out-tree $T = (V_T, E_T)$ rooted at v with exactly k' internal nodes and l' leaves, such that $V_T = U$. This claim implies that any multilinear monomial of $P_{v,k',l'}$ is of degree exactly $k' + l'$.

The claim is clearly true for the base cases, and thus we next assume that $k' > 0$, and the claim is true for all $v \in V$, $k^* \in \{0, ..., k'\}$ and $l^* \in \{1, ..., l'\}$, such that ($k^* < k'$ or $l^* < l'$).

Let $\prod_{w \in U} x_w$, for some $U \subseteq V$, be a monomial of $P_{v,k',l'}$. By the definition of $C_{v,k',l'}$, there is u such that $(v, u) \in E$, for which we have the following cases.

1. If $P_{u,k'-1,l'}$ has a monomial $\prod_{w \in U \setminus \{v\}} x_w$: By the induction hypothesis, G has an out-tree $T_u = (V_u, E_u)$ rooted at u with exactly $k' - 1$ internal nodes and l' leaves, such that $V_u = U \setminus \{v\}$. By adding v and (v, u) to T_u, we get an out-tree $T = (V_T, E_T)$ of G that is rooted at v, has exactly k' internal nodes and l' leaves, and such that $V_T = U$.
2. Else: There are $k^* \in \{1, ..., k'\}$, $l^* \in \{1, ..., l' - 1\}$ and $U^* \subseteq U$, such that P_{v,k^*,l^*} has the monomial $\prod_{w \in U^*} x_w$, and $P_{u,k'-k^*,l'-l^*}$ has the monomial $\prod_{w \in U \setminus U^*} x_w$. By the induction hypothesis, G has an out-tree $T_v = (V_v, E_v)$ rooted at v with exactly k^* internal nodes and l^* leaves, such that $V_v = U^*$. Moreover, G has an out-tree $T_u = (V_u, E_u)$ rooted at u with exactly $k' - k^*$ internal nodes and $l' - l^*$ leaves, such that $V_u = U \setminus U^*$. Thus, we get that the out-tree $T = (U, E(T_v) \cup E(T_u) \cup (v, u))$ of G is rooted at v, and has exactly k' internal nodes and l' leaves.

We get that G has an out-tree rooted at r of exactly k internal nodes and l leaves iff $P_{r,k,l}$ has a mutlilinear monomial of degree at most t. □

The definition of $(C_{r,k,l}, X, t)$ immediately implies the following observation.

Observation 2. *We can compute $(C_{r,k,l}, X, t)$ in polynomial time and space.*

2.3 The Algorithm IOB-Alg[Tree-Alg]

Koutis et al. [7,17] gave an $O^*(2^t)$ time and polynomial space randomized algorithm for t-MLD. We denote this algorithm by MLD-Alg, and use it to get an algorithm for (k,l)-Tree (see Algorithm 2).

Algorithm 2. Tree-Alg(G, r, k, l)

1: Compute $f(G, r, k, l) = (C_{r,k,l}, X, t)$.
2: Accept iff MLD-Alg$(C_{r,k,l}, X, t)$ accepts.

By Lemmas 1 and 2, and Observation 2, we have the following theorem.

Theorem 1. IOB-Alg[Tree-Alg] *is an $O^*(4^k)$ time and polynomial space randomized algorithm for k-IOB.*

3 A k-IOB Algorithm for Graphs of Bounded Degree Δ

3.1 A Modification of the Algorithm IOB-Alg[A]

We first prove that in Step 3 of IOB-Alg[A] (see Section 2.1), we can iterate over less than k values for l.

Given an out-tree $T = (V_T, E_T)$ and $i \in \mathbb{N}$, denote the number of degree-i nodes in T by n_i^T.

Observation 3 ([14]). *If $|V_T| \geq 2$, then $2 + \sum_{3 \leq i}(i-2)n_i^T = n_1^T$.*

Observation 4. *An out-tree T of G with exactly k internal nodes contains an out-tree with exactly k internal nodes and at most $k - \frac{k-2}{\Delta-1}$ leaves.*

Proof. As long as T has an internal node v with at least two out-neighbors that are leaves, delete one of these leaves and its adjacent edge from T. Denote the resulting out-tree by T', and denote the tree that we get after deleting all the leaves in T' by T''. Note that T' has exactly k internal nodes, and that T' and T'' have the same number of leaves. Since T'' has k nodes and bounded degree Δ, Observation 3 implies that if $n_1^{T''} + n_\Delta^{T''} = k$, then $n_1^{T''} = k - \frac{k-2}{\Delta-1}$, and if $n_1^{T''} + n_\Delta^{T''} < k$, then $n_1^{T''} < k - \frac{k-2}{\Delta-1}$. We have that $n_1^{T''} \leq k - \frac{k-2}{\Delta-1}$, and thus we conclude that T' has exactly k internal nodes and at most $k - \frac{k-2}{\Delta-1}$ leaves. □

Thus, in Step 3 of IOB-Alg[A], we can iterate only over $l = 1, 2, ..., k - \lceil \frac{k-2}{\Delta-1} \rceil$. We add some preprocessing steps to IOB-Alg[A], and thus get Δ-IOB-Alg[A] (see Algorithm 3). These preprocessing steps will allow us to assume, when presenting algorithm A, that the underlying undirected graph of G is a connected graph that is neither a cycle nor a clique. This assumption will allow us to compute a proper Δ-coloring of the underlying undirected graph of G (see Section 3.3), which we will use in the following Section 3.2.

Algorithm 3. Δ-IOB-Alg[A](G, k)

1: **if** $k \geq |V|$ or the underlying undirected graph of G is not connected **then**
2: Reject.
3: **else if** the underlying undirected graph of G is a cycle **then**
4: **if** $k = |V| - 1$ **then** Accept iff G has a hamiltonian path. **else** Accept iff there is at most one node of out-degree 2 in G. **end if**
5: **else if** the underlying undirected graph of G is a clique **then**
6: Accept.
7: **end if**
8: **for all** $r \in V$ **do**
9: **if** G has no out-branching T rooted at r **then** Go to the next iteration. **end if**
10: **for** $l = 1, 2, ..., k - \lceil \frac{k-2}{\Delta-1} \rceil$ **do**
11: **if** A(G, r, k, l) accepts **then** Accept. **end if**
12: **end for**
13: **end for**
14: Reject.

We can clearly perform the new preprocessing steps in $O(|E|)$ time and $O(|V|)$ space. Steps 2 and 4 are clearly correct. Since a tournament (i.e., a directed graph obtained by assigning a direction for each edge in an undirected complete graph) has a hamiltonian path [13], we have that Step 6 is also correct.

We have the following lemma.

Lemma 3. *Δ-IOB-Alg[A] is an $O(\sum_{r \in V}(|E| + \sum_{1 \leq l \leq k - \lfloor \frac{k-2}{\Delta-1} \rfloor} t(G, r, k, l)))$ time and $O(|V| + \max_{r \in V, 1 \leq l \leq k - \lceil \frac{k-2}{\Delta-1} \rceil} s(G, r, k, l))$ space algorithm for k-IOB.*

3.2 A Modification of the Reduction f

In this section assume that we have a proper Δ-coloring $col : V \rightarrow \{c_1, ..., c_\Delta\}$ of the underlying undirected graph of G. Having such col, we modify the reduction f (see Section 2.2) to construct a "better" input for t-MLD (i.e., an input in which $t < k + l$).

The Idea Behind the Modification: Recall that in the previous construction, we let each monomial represent a certain pair of an out-tree $T = (V_T, E_T)$ and a function $h : V_T \rightarrow V$. The monomial included indeterminates representing *all* the nodes to which the nodes in V_T are mapped. We can now select some color $c \in \{c_1, ..., c_\Delta\}$, and ignore some occurrences of indeterminates that represent nodes whose color is c and whose degree in $h(T)$ is Δ. We thus construct monomials with smaller degrees, and have an input for t-MLD in which $t < k + l$.

More precisely, the monomial representing T and h is $\prod_{v \in U} x_{h(v)}$, where U is V_T, excluding nodes mapped to nodes whose color is c and whose degree in T is Δ (except the root). We add constraints on T and h to garauntee that nodes in V_T that are mapped to the same node do not have common neighbors in T.

The correctness is based on the following observation. Suppose that there is an indeterminate x_v that occurs more than once in the original monomial representing T and h, but not in the new monomial representing them. Thus the color of v is c. Moreover, there are different nodes $u, w \in V_T$ such that $h(u) = h(w) = v$, and the degree of u in T is Δ. We get that u has a neighbor u' in T and w has a *different* neighbor w' in T, such that $h(u') = h(w')$ and the color of $h(u')$ is not c. Thus $x_{h(u')}$ occurs more than once in the new monomial representing T and h. This implies that monomials that are not multilinear in the original construction do not become multilinear in the new construction.

The Construction: Let (G, r, k, l) be an input for (k, l)-Tree. We now construct an input $f(G, r, k, l, col) = (C, X, t)$ for t-MLD.

We add a node r' to V and the edge (r', r) to E. We color r' with some $c \in \{c_1, ..., c_\Delta\} \setminus \{col(r)\}$. In the following let $<$ be some order on $V \cup \{nil\}$, such that nil is the smallest element. Define $X = \{x_v : v \in V\}$, and $t = (2 - \frac{\Delta+1}{\Delta(\Delta-1)})k + 8$. Denote $N(v, i, o) = \{u \in V \setminus \{i\} : (v, u) \in E, u > o\}$.

We inductively define an arithmetic circuit $C^{c,i,o,b}_{v,k',l'}$ over X, for all $v \in V, k' \in \{0, ..., k\}$, $l' \in \{1, ..., l\}$, $c \in \{c_1, ..., c_\Delta\}$, i such that $(i, v) \in E$, o such that $(v, o) \in E$ or $o = nil$, and $b \in \{F, T\}$. Informally, v, k' and l' play the same role as in the original construction; c indicates that only indeterminates representing nodes colored by c can be ignored; i and o are used for constraining the pairs of trees and functions represented by monomials as noted in "The Idea Behind the Modification"; and b indicates whether the indeterminate of v is ignored.

Base Cases:

1. If $k' = 0$, $l' = 1$ and $b = F$: $C^{c,i,o,b}_{v,k',l'} = x_v$.
2. Else if $[k' = 0]$ or $[N(v, i, o) = \emptyset]$ or $[(|N(v, i, o)| > l'$ or $col(v) \neq c$ or $v = r$ or $|N(v, i, nil)| < \Delta - 1)$ and $b = T]$: $C^{c,i,o,b}_{v,k',l'} = 0$.

Steps: (assume that none of the base cases applies)

1. If $l' = 1$ and $b = F$: $C^{c,i,o,b}_{v,k',l'} = x_v \sum_{u \in N(v,i,o)} (C^{c,v,nil,F}_{u,k'-1,l'} + C^{c,v,nil,T}_{u,k'-1,l'})$.
2. Else if $b = F$:
$$C^{c,i,o,b}_{v,k',l'} = \sum_{u \in N(v,i,o)} [x_v C^{c,v,nil,F}_{u,k'-1,l'} + x_v C^{c,v,nil,T}_{u,k'-1,l'} + \sum_{1 \leq k^* \leq k'} \sum_{1 \leq l^* \leq l'-1} C^{c,i,u,b}_{v,k^*,l^*} (C^{c,v,nil,F}_{u,k'-k^*,l'-l^*} + C^{c,v,nil,T}_{u,k'-k^*,l'-l^*})].$$
3. If $b = T$ and there is exactly one node u in $N(v, i, o)$: $C^{c,i,o,b}_{v,k',l'} = C^{c,v,nil,F}_{u,k'-1,l'}$.
4. Else if $b = T$:
   (a) Denote $u = \min(N(v, i, o))$.
   (b) $C^{c,i,o,b}_{v,k',l'} = \sum_{1 \leq k^* \leq k'} \sum_{1 \leq l^* \leq l'-1} C^{c,i,u,b}_{v,k^*,l^*} C^{c,v,nil,F}_{u,k'-k^*,l'-l^*}$.

The following order shows that when computing an arithmetic circuit $C^{c,i,o,b}_{v,k',l'}$, we only use arithmetic circuits that have been already computed.

Order:

1. For $k' = 0, 1, ..., k$:
   (a) For $l' = 1, 2, ..., l$:
      i. $\forall v \in V, c \in \{c_1, ..., c_\Delta\}$, i s.t. $(i, v) \in E$, o s.t. $(v, o) \in E$ or $o = nil$, $b \in \{F, T\}$: Compute $C^{c,i,o,b}_{v,k',l'}$.

Define $C = \sum_{c\in\{c_1,...,c_\Delta\}} C^{c,r',nil,F}_{r,k,l}$.

Denote the polynomial that $C^{c,i,o,b}_{v,k',l'}$ (resp. C) represents by $P^{c,i,o,b}_{v,k',l'}$ (resp. P).

Correctness: We need the next two definitions, which we illustrate in Fig. 1.

Definition 1. *Let $v \in V$, $k' \in \{0,...,k\}$, $l' \in \{1,...,l\}$, $c \in \{c_1,...,c_\Delta\}$, i such that $(i,v) \in E$, o such that $(v,o) \in E$ or $o = nil$. Given a* subgraph $T = (V_T, E_T)$ *of G, we say that*

1. *T is a (v,k',l',c,i,o,F)-tree if*
   (a) *T is an out-tree rooted at v with exactly k' internal nodes and l' leaves.*
   (b) *Every out-neighbor of v in T belongs to $N(v,i,o)$.*
2. *T is a (v,k',l',c,i,o,T)-tree if*
   (a) *$col(v) = c$, $v \neq r$, and $|N(v,i,nil)| = \Delta - 1$.*
   (b) *Every node in $N(v,i,o)$ is an out-neighbor of v in T, and $N(v,i,o) \neq \emptyset$.*
   (c) *There is at most one node $i' \in V_T$ such that $(i',v) \in E_T$.*
      i. *If such an i' exists: $(v,i') \notin E_T$, and $T' = (V_T, E_T \setminus \{(i',v)\})$ is an out-tree rooted at v.*
      ii. *Else: T is a (v,k',l',c,i,o,F)-tree.*

Definition 2. *Given a (v,k',l',c,i,o,b)-tree $T = (V_T, E_T)$, define $I(T) =$*

$$\{u \in V_T : [u \neq v \wedge (col(u) \neq c \vee u \text{ has less than } (\Delta-1) \text{ out}-\text{neighbors in } T)]$$
$$\vee [u = v \wedge (b = F \vee v \text{ has an in}-\text{neighbor in } T)]\}.$$

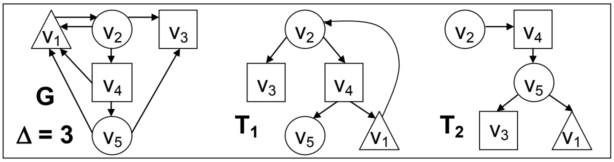

Fig. 1. Assume that $r = v_1 < v_2 < v_3 < v_4 < v_5$, and that shapes represent colors. We have that T_1 is a (v_2,k',l',O,v_1,nil,T)-tree for any k' and l', and $I(T_1) = \{v_1,v_2,v_3,v_4,v_5\}$. Moreover, T_2 is a $(v_2,3,2,O,v_1,v_3,T)$-tree, and $I(T_2) = \{v_1,v_3,v_4\}$.

Observation 5. *Let $T = (V_T, E_T)$ be a (v,k',l',c,i,o,b)-tree of G, such that there is no $i' \in V_T$ for which $(i',v) \in E_T$. Then, $P^{c,i,o,b}_{v,k',l'}$ has the (multilinear) monomial $\prod_{w\in I(T)} x_w$.*

Proof. We prove the claim by using induction on the construction. The claim is clearly true for the base cases. Next consider a (v,k',l',c,i,o,b)-tree $T = (V_T, E_T)$ of G, such that $C^{c,i,o,b}_{v,k',l'}$ is not constructed in the base cases. Assume that the claim is true for all $(\widetilde{v},\widetilde{k},\widetilde{l},\widetilde{c},\widetilde{i},\widetilde{o},\widetilde{b})$ such that $C^{\widetilde{c},\widetilde{i},\widetilde{o},\widetilde{b}}_{\widetilde{v},\widetilde{k},\widetilde{l}}$ is constructed before $C^{c,i,o,b}_{v,k',l'}$. Denote by u the smallest out-neighbor of v in T.

Denote by $T_v = (V_v, E_v)$ and $T_u = (V_u, E_u)$ the two out-trees of G in the forest $F = (V_T, E_T \setminus \{(v, u)\})$, such that $v \in V_v$. If $u \notin I(T)$ (this is not the case if $b = T$, since then $col(u) \neq c$), then denote $b' = T$, and note that the set of out-neighbors of u in T_u contains all of the neighbors of u in G, excluding v; else denote $b' = F$. We have the following cases.

1. If $|V_v| = 1$: T_u is a $(u, k'-1, l'c, v, nil, b')$-tree of G. If $b = F$, then $I(T_u) = I(T) \setminus \{v\}$; else $I(T_u) = I(T_v)$. By the induction hypothesis $C^{c,v,nil,b'}_{u,k'-1,l'}$ has the monomial $\prod_{w \in I(T_u)} x_w$. Thus, by the definition of $C^{c,i,o,b}_{v,k',l'}$, $P^{c,i,o,b}_{v,k',l'}$ has the required monomial.
2. Else: Denote the number of internal nodes and leaves in T_v by k_v and l_v, respectively. Note that $1 \leq k_v \leq k'$, $1 \leq l_v < l'$, T_v is a $(v, k_v, l_v, c, i, u, b)$-tree of G, and T_u is a $(u, k'-k_v, l'-l_v, c, v, nil, b')$-tree of G. Moreover, $I(T_v)$ and $I(T_u)$ are disjoint sets whose union is $I(T)$. By the induction hypothesis, $P^{c,i,u,b}_{v,k_v,l_v}$ has the monomial $\prod_{w \in I(T_v)} x_w$, and $P^{c,v,nil,b'}_{u,k'-k_v,l'-l_v}$ has the monomial $\prod_{w \in I(T_u)} x_w$. By the definition of $C^{c,i,o,b}_{v,k',l'}$, $P^{c,i,o,b}_{v,k',l'}$ has the monomial $\prod_{w \in I(T_v)} x_w \prod_{w \in I(T_u)} x_w = \prod_{w \in I(T)} x_w$.

□

Observation 6. *If $P^{c,i,o,b}_{v,k',l'}$ has a (multilinear) monomial $\prod_{w \in U} x_w$, for some $U \subseteq V$, then G has a (v, k', l', c, i, o, b)-tree T such that $I(T) = U$.*

Proof. We prove the claim by using induction on the construction. The claim is clearly true for the base cases. Let $\prod_{w \in U} x_w$, for some $U \subseteq V$, be a monomial of $P^{c,i,o,b}_{v,k',l'}$, such that $C^{c,i,o,b}_{v,k',l'}$ is not constructed in the base cases. Assume that the claim is true for all $C^{\widetilde{c},\widetilde{i},\widetilde{o},\widetilde{b}}_{\widetilde{v},\widetilde{k},\widetilde{l}}$ that is constructed before $C^{c,i,o,b}_{v,k',l'}$.

First suppose that $b = F$. By the definition of $C^{c,i,o,b}_{v,k',l'}$, there are $u \in N(v, i, o)$ and $b' \in \{F, T\}$ such that one of the next conditions is fulfilled.

1. $C^{c,v,nil,b'}_{u,k'-1,l'}$ has the monomial $\prod_{w \in U \setminus \{v\}} x_w$. By the induction hypothesis, G has a $(u, k'-1, l', c, v, nil, b')$-tree $T_u = (V_u, E_u)$, such that $I(T_u) = U \setminus \{v\}$. Suppose that there is $i' \in V_u$ such that $(i', u) \in E_u$. In this case $b' = T$; thus $v \notin V_u$ and the set of out-neighbors of u in T_u contains all the neighbors of u in G, excluding v. We get that i' is an out-neighbor of u in T_u, which a contradiction. Thus, by adding v and (v, u) to T_u, we get a (v, k', l', c, i, o, b)-tree T such that $I(T) = U$ (since $I(T) = I(T_u) \cup \{v\}$).
2. There are $k^* \in \{1, ..., k'\}$, $l^* \in \{1, ..., l'-1\}$ and $U^* \subseteq U$, such that $P^{c,i,u,b}_{v,k^*,l^*}$ has the monomial $\prod_{w \in U^*} x_w$, and $P^{c,v,nil,b'}_{u,k'-k^*,l'-l^*}$ has the monomial $\prod_{w \in U \setminus U^*} x_w$. By the induction hypothesis, G has a $(v, k^*, l^*, c, i, u, b)$-tree $T_v = (V_v, E_v)$ such that $I(T_v) = U^*$, and a $(u, k'-k^*, l'-l^*, c, v, nil, b')$-tree $T_u = (V_u, E_u)$ such that $I(T_u) = U \setminus U^*$. Consider the following cases.
   (a) If $v \in V_u$: $v \notin I(T_u)$ (since $v \in I(T_v)$). Thus $col(v) = c$ and v has $\Delta - 1$ out-neighbors in T_u. Note that v is not an out-neighbor of u in T_u, and thus u is an out-neighbor of v in T_u. Therefore $b' = T$, and thus $col(u) = c$, which is a contradiction (since col is a proper coloring).

(b) If there is $w \in (V_v \cap V_u) \setminus \{v, u\} \neq \emptyset$: Since $I(T_v) \cap I(T_u) = \emptyset$, we get that $col(w) = c$ and (w has Δ neighbors in T_v or T_u). Thus there is w' that is a neighbor of w in both T_v and T_u, such that $col(w') \neq c$. We get that $w' \in I(T_v) \cap I(T_u) = \emptyset$, which is a contradiction.

(c) If $u \in V_v$: u is not an out-neighbor of v in T_v. Therefore u has less than $\Delta - 1$ out-neighbors in T_v, and thus $u \in I(T_v)$. We get that $u \notin I(T_u)$, which implies that the set of out-neighbors of u in T_u contains all the neighbors of u in G, excluding v. Thus u has a neighbor, which is not v, in both T_v and T_u, and we have a contradiction according to Case 2b.

We get that $V_v \cap V_u = \emptyset$. If there is $i' \in V_u$ such that $(i', u) \in E_u$, then we get a contradiction in the same manner as in Case 1. We get that $T = (V_v \cup V_u, E_v \cup E_u \cup \{(v, u)\})$ is an out-tree of G. It is straightforward to verify that T is a (v, k', l', c, i, o, b)-tree of G such that $I(T) = I(T_v) \cup I(T_u)$ (and thus $I(T) = U$).

Now suppose that $b = T$. Denote by u the smallest node in $N(v, i, o)$. By the definition of $C^{c,i,o,b}_{v,k',l'}$, one of the next conditions is fulfilled.

1. If $N(v, i, o) = \{u\}$: $P^{c,v,nil,F}_{u,k'-1,l'}$ has the monomial $\prod_{w \in U} x_w$. By the induction hypothesis, G has a $(u, k' - 1, l', c, v, nil, F)$-tree T_u such that $I(T_u) = U$. Since v is not an out-neighbor of u in T_u, by adding v and (v, u) to T_v, we get a (v, k', l', c, i, o, b)-tree T of G (which may not be an out-tree), such that $I(T) = I(T_u) = U$.
2. Else: There are $k^* \in \{1, ..., k'\}$, $l^* \in \{1, ..., l' - 1\}$ and $U^* \subseteq U$, such that $P^{c,i,u,b}_{v,k^*,l^*}$ has the monomial $\prod_{w \in U^*} x_w$, and $P^{c,v,nil,F}_{u,k'-k^*,l'-l^*}$ has the monomial $\prod_{w \in U \setminus U^*} x_w$. By the induction hypothesis, G has a $(v, k^*, l^*, c, i, u, b)$-tree $T_v = (V_v, E_v)$ such that $I(T_v) = U^*$, and a $(u, k' - k^*, l' - l^*, c, v, nil, F)$-tree $T_u = (V_u, E_u)$ such that $I(T_u) = U \setminus U^*$. Consider the following cases.

(a) If there is $w \in (V_v \cap V_u) \setminus \{v, u\} \neq \emptyset$: We get a contradiction in the same manner as in the previous Case 2b.

(b) If $u \in V_v$: Since $col(u) \neq c$, we get that $u \in I(T_v) \cup I(T_u) = \emptyset$, which is a contradiction.

We get that $V_v \cap V_u \setminus \{v\} = \emptyset$. Denote $T = (V_T = (V_v \cup V_u), E_T = (E_v \cup E_u \cup \{(v, u)\}))$. Suppose, by way of contradiction, that there are two nodes $i_1, i_2 \in V_T$ such that $(i_1, v), (i_2, v) \in E_T$. Since T_v is a $(v, k^*, l^*, c, i, u, b)$-tree and T_u is an out-tree, we can assume WLOG that $i_1 \in V_v$ and $i_2 \in V_u$. We get that $v \in I(T_v)$, and thus $v \notin I(T_u)$. Therefore v has $\Delta - 1$ out-neighbors in T_u; but since T_u is an out-tree rooted at u, and v is not an out-neighbor of u in T_u, we have a contradiction. Thus we get that T is a (v, k', l', c, i, o, b)-tree of G such that $I(T) = I(T_v) \cup I(T_u)$ (and thus $I(T) = U$). □

Observation 7. *If (G, r, k, l) has a solution, then P has a multilinear monomial of degree at most t.*

Proof. Let $T = (V_T, E_T)$ be a solution. Denote $n(T, c) = \{v \in V_T : col(v) = c, v \text{ has } \Delta \text{ neighbors in } T\}$, and $c^* = \text{argmax}_{c \in \{c_1, \ldots, c_\Delta\}}\{|n(T, c)|\}$. By Observation 4 and the pseudocode of Δ-IOB-Alg[A] (see Section 3.1), we get that

1. $2 + \sum_{3\leq i\leq \Delta}(i-2)n_i^T = n_1^T$.
2. $\sum_{1\leq i\leq \Delta} n_i^T = k + l$.
3. $n_1^T - 1 \leq l \leq k - \frac{k-2}{\Delta-1}$.
4. $|n(T,c^*)| \geq n_\Delta^T/\Delta$.

These conditions imply that $k+l-|n(T,c^*)| \leq (2-\frac{\Delta+1}{\Delta(\Delta-1)})k+7$. Since T is an (r,k,l,c^*,r',nil,F)-tree, the definition of C and Observation 5 imply that P has the (multilinear) monomial $\prod_{w\in I(T)} x_w$. Note that $|I(T)| \leq k+l-|n(T,c^*)|+1$, and thus we get the observation. □

Since Observation 6 implies that if P has a multilinear monomial, then (G,r,k,l) has a solution, and by Observation 7, we get the following lemma.

Lemma 4. *(G,r,k,l) has a solution iff (C,X,t) has a solution.*

The definition of (C,X,t) immediately implies the following observation.

Observation 8. *We can compute (C,X,t) in polynomial time and space.*

3.3 The Algorithm Δ-IOB-Alg[Δ-Tree-Alg]

Skulrattanakulchai [16] gave a linear-time algorithm that computes a proper Δ-coloring of an undirected connected graph of bounded degree Δ, which is not an odd cycle or a clique. In Δ-Tree-Alg (see Algorithm 4), we assume that the underlying undirected graph of G is connected, and that it is not a cycle or a clique, since these cases are handled in the preprocessing steps of Δ-IOB-Alg[A].

Algorithm 4. Δ-Tree-Alg(G,r,k,l)

1: Use the algorithm in [16] to get a proper Δ-coloring *col* of the underlying undirected graph of G.
2: Compute $f(G,r,k,l,col) = (C,X,t)$.
3: Accept iff MLD-Alg(C,X,t) accepts.

By Lemmas 3 and 4, and Observation 8, we have the following theorem.

Theorem 2. Δ-IOB-Alg[Δ-Tree-Alg] *is an $O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})$ time and polynomial space randomized algorithm for k-IOB.*

4 Open Questions

In this paper we have presented an $O^*(4^k)$ time algorithm for k-IOB, which improves the previous best known O^* running time for k-IOB. However, our algorithm is randomized, while the algorithm that has the previous best known O^* running time is deterministic. Can we obtain an $O^*(4^k)$ time deterministic algorithm for k-IOB? Moreover, can we further reduce the $O^*(4^k)$ and $O^*(2^{(2-\frac{\Delta+1}{\Delta(\Delta-1)})k})$ running times for k-IOB presented in this paper?

References

1. Cohen, N., Fomin, F.V., Gutin, G., Kim, E.J., Saurabh, S., Yeo, A.: Algorithm for finding k-vertex out-trees and its application to k-internal out-branching problem. J. Comput. Syst. Sci. 76(7), 650–662 (2010)
2. Demers, A., Downing, A.: Minimum leaf spanning tree. US Patent no. 6,105,018 (August 2013)
3. Fomin, F.V., Gaspers, S., Saurabh, S., Thomassé, S.: A linear vertex kernel for maximum internal spanning tree. J. Comput. Syst. Sci. 79(1), 1–6 (2013)
4. Fomin, F.V., Grandoni, F., Lokshtanov, D., Saurabh, S.: Sharp separation and applications to exact and parameterized algorithms. Algorithmica 63(3), 692–706 (2012)
5. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified NP-complete problems. In: Proc. STOC, pp. 47–63 (1974)
6. Gutin, G., Razgon, I., Kim, E.J.: Minimum leaf out-branching and related problems. Theor. Comput. Sci. 410(45), 4571–4579 (2009)
7. Koutis, I.: Faster algebraic algorithms for path and packing problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
8. Koutis, I., Williams, R.: Limits and applications of group algebras for parameterized problems. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 653–664. Springer, Heidelberg (2009)
9. Nederlof, J.: Fast polynomial-space algorithms using mobius inversion: improving on steiner tree and related problems. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 713–725. Springer, Heidelberg (2009)
10. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford University Press (2006)
11. Ozeki, K., Yamashita, T.: Spanning trees: A survey. Graphs and Combinatorics 27(1), 1–26 (2011)
12. Prieto, E., Sloper, C.: Reducing to independent set structure – the case of k-internal spanning tree. Nord. J. Comput. 12(3), 308–318 (2005)
13. Rédei, L.: Ein kombinatorischer satz. Acta Litteraria Szeged 7, 39–43 (1934)
14. Raible, D., Fernau, H., Gaspers, D., Liedloff, M.: Exact and parameterized algorithms for max internal spanning tree. Algorithmica 65(1), 95–128 (2013)
15. Salamon, G.: A survey on algorithms for the maximum internal spanning tree and related problems. Electronic Notes in Discrete Mathematics 36, 1209–1216 (2010)
16. Skulrattanakulchai, S.: Delta-list vertex coloring in linear time. Inf. Process. Lett. 98(3), 101–106 (2006)
17. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. Inf. Process. Lett. 109(6), 315–318 (2009)